SOUND SYNTHESIS FOR NONLINEARLY DEFORMABLE BODIES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Steven Shi An

January 2013

SOUND SYNTHESIS FOR NONLINEARLY DEFORMABLE BODIES

Steven Shi An, Ph.D.

Cornell University 2013

The computer graphics community has made great advances in rendering the visual appearance and dynamic motion of real world phenomena, but relatively little has been done to render the sounds we hear with the same amount of plausibility and realism. While pre-recorded sounds may give the best results for pre-rendered movies, this is not a satisfying approach for real-time, interactive applications. In order to provide plausible sounds for interactive virtual environments, automated methods synchronized to visual simulations must be developed. In the past couple of decades, there has been increasing interest within and outside the graphics community to tackle this problem, and many techinques, such as linear modal models for rigid body sounds, have been successfully developed. This thesis presents new methods which advance the state of the art by making automated, realistic sound synthesis more practical for a wider range of physical phenomena. In particular, it introduces techniques for two types of nonlinearly deformable bodies: thin shells and cloth.

Thin shell objects, such as trash cans and plastic water containers, do not deform very much visually, so their appearance and motion may be well approximated as rigid bodies. However, they are still significantly more flexible than solid objects, and this greatly affects the sounds they produce. Linear modal sound models are no longer sufficient. Due to the nonlinaer nature of their deformations, the modes do not vibrate independently, and thus a more sophisticated dynamics model is needed to integrate their audible motion. This thesis first introduces *cubature optimization* as a method for simulating

nonlinear reduced deformable models in general, and then it shows how it can be used to enhance modal models for practical synthesis of thin shell sounds. Beyond the computational costs, modal models also require rather large memory costs at run-time, as all vibrational modes must be accessible. These modes can be compressed by very large amounts, and methods for doing so are presented as well. Finally, another type of highly deformable body is cloth. Cloth has been well-studied by the computer graphics community, and this thesis presents a practical method for synthesizing sounds synchronized to cloth animaitons. Unlike the case of thin shells, this method is largely data-driven, as cloth is a much more complex material. Existing cloth simulation techniques, while successful in producing convincing visual motion, cannot practically model all the subtleties of cloth structure which greatly affect the resulting sound. Instead, we use a database of recordings with a motion-driven synthesis pipeline to synthesize plausible, natural-sounding results.

BIOGRAPHICAL SKETCH

Steven Shi An was born on February 13th, 1984 in Shenyang, China. He came to the US in 1989 and lived in various cities. After graduating from Palo Alto High School in California, he attended UC Berkeley and earned a BS in Electrical Engineering and Computer Science. He then worked for a year at a video game studio before starting his PhD studies at Cornell University in 2007.

This document is dedicated to my parents, who have worked extremely hard to support me in my pursuits and provide me with opportunities in life they never had. I love them both deeply and am eternally grateful to have such great parents.

ACKNOWLEDGEMENTS

I am extremely grateful to all the amazing people whom I have had the pleasure of meeting during my five years at Cornell. You have all affected my life in a very positive way, and without you, I would not be as fulfilled as I am today.

Of course, I must first thank my advisor, Doug James, for the challenging problems and luxurious support he has provided throughout my PhD career. He has taught me a great deal about research, and he has re-defined the meaning of "impossible" for me. Thanks to his optimism and relentless drive, I have solved some ridiculously difficult problems that I originally thought were hopeless. His high standards for everything we produce, from the actual research to the formatting of our presentation slides, will stay with me for the rest of my career. Without his guidance and knowledge, I would have none of the accomplishments presented in this thesis.

I am also very grateful for my prestigious committee members, Charlie van Loan and Dan Huttenlocher. I consider Charlie to be one of the greatest lecturers and speakers I have ever met, and his class on matrix computations is one of the most useful classes I have ever taken. I have the upmost respect for Dan, and I see him as a role model for straddling the fields of computer science and business throughout his career.

I also wish to thank everyone who has collaborated with me on projects and papers. Ted Kim was crucial to the success of my first paper, and I wish him the best of luck at UC Santa Barbara. Jeff Chadwick was the first author on my next paper, and I could not imagine a more reliable and studious researcher. He has also been a fantastic roommate for most of my time in Ithaca, and I am sure he will go on to do great things in the future. Not to mention, he is a fantastic musician. I also must thank Changxi Zheng for a variety of reasons, including providing me with his clean code-base, helping me

with the compression project, and just for his overall positive attitude. I wish him the best of luck at Columbia, where I am sure he will continue to do fascinating research. Steve Marschner was a co-author on my last paper, and I was also a TA for his graphics class for one semester. Steve was always a pleasure to work with, and he provided many crucial insights as well as moral support when things got tough. I also TA'd for Noah Snavely, a relatively new professor who will most certainly become an influential force in machine vision, who also provided some key insights for the cloth sounds project. Some other people who gratiously offered their help on that project and others include Sean Chen, Carol Krumhansl, Taylan Cihan, Tianyu Wang, Anna Blasiak, and Kat Agres. I can only hope to one day repay them for their time and generosity. Lastly, I wish to thank Kavita Bala for her help on the compression project and for just being a positive force in the graphics group.

I would also like to thank the other students and post-docs in the graphics program for their companionship and help, including Jon Kaldor, Cem Yuskel, Wenzel Jacob, Tim Condon, Jon Moon, and Shuang Zhao. Throughout the years and especially before SIGGRAPH deadlines, I would not have been able to keep my sanity without them being in the thick of it with me. I wish them all the best of luck, and it was an honor to study in the same program with them.

Lastly, I would like to thank all the friends that I have made during my time at Cornell. They are too numerous to list, but you know who you are. I have grown so much as an individual, both personally and professionally, and you have all been such a huge influence on me. Many of you I met in the computer science department, some through the GPSA, EGSA, and SAAGA, some through random hobbies, and even more through social gatherings. Thanks to you all, I can confidently say that my time in Ithaca will be one of the most memorable periods of my life. I wish you all the best of luck, and I am

sure I will run into you again at some point in the future.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

One of the core goals of computer graphics research is to render convincing, interactive virtual environments. This has a wide range of applications, such as computer-generated movies, video games, teleconferencing, and training simulations for a variety of industries. Understandably, the community has largely been motivated by the visual aspects of this problem since its inception. For many decades, much work has been done in simulating the behavior of light to produce realistically lit scenes. The physical motion of objects, such as rigid bodies, cloth, and fluids, has also been given much attention. Thanks to recent advances in hardware and algorithms, current systems achieve impressively realistic simulations at interactive speeds. Lastly, giving content creators tools to quickly and intuitively model the appearance and behavior of objects has been an important focus of computer graphics as well.

However, the visual system is certainly not the only way in which humans experience the world. Our sense of hearing, touch, and even smell all contribute in important ways. If the goal is to create more convincing virtual environments, the community must then address these other modes of sensing as well. Relatively recently, the problem of sound synthesis and rendering has gained increasing attention from researchers, and it is the primary focus of this thesis. This chapter will give an overview of prior work in sound synthesis, and the remaining chapters present novel sound synthesis techniques for non-linearly deformable bodies. Subsequent chapters also present prior work specifically relevant to them.

1.1 Background

Sound is fundamentally caused by vibrations in a medium, such as air and water, surrounding a human listener. If the vibrations are within the audible frequency range, which depends on the listener but is typically accepted as 20 Hz to 20 kHz, then the human ear interprets the vibrations as sound. On a computer system, sound can be produced via speaker or headphones for the user to experience. The role of *sound rendering* is to control the speakers so they produce the proper sounds for a *virtual listening location*. In other worlds, for a virtual character in a virtual world, the goal is to compute what audible medium vibrations the ears of the virtual character will hear and then forward them to the human user via speakers. If the speakers have stereo functionality, the system may also produce slightly different signals for each channel, one from the virtual left and another from the right, to provide spatialization cues.

The sound rendering pipeline [131] can be viewed as having two major stages: the first is *sound synthesis*, which focuses on modeling what sounds are emitted from a given object or event in the environment at a given time. The second stage is *sound propagation*, which models how sound waves propagate throughout a given environment, ultimately arriving at the ears of the virtual listener. The focus of this thesis is on the *synthesis* stage, and we refer the reader to papers such as [113, 67, 136] for further background on propagation.

The remaining sections provide an overview of prior work in sound synthesis for a variety for physical phenomena, from rigid body impacts to fire combustion to foot steps. While techniques for general sound synthesis (such as for music and speech) are useful and relevant, this chapter is focused more on methods designed to accompany physics-

based animations and virtual environments. It is helpful to organize such methods on a spectrum from physics-based (Section 1.2) to data-driven (Section 1.3), and some which are better described as a hybrid of the two (Section 1.3.3). Note that in some sound synthesis literature, the term "physics-based" is used to describe some of these data-driven methods as well. In this thesis, it will be used to refer to methods which derive sound characteristics and models purely from physical and mechanical models, not from recordings.

1.2 Prior Work: Physics-Based Methods

Physics-based sound synthesis techniques predict acoustic emissions from a given object or phenomena by modeling the underlying physical systems and how they affect the surrounding medium. The primary advantage of such techniques is that they automatically provide plausible variety, synchronized with the physical simulation, without any reliance on recorded sounds. Manual intervention is also minimal, and it is typically only required for tuning parameters in order to get desirable results. Techniques vary in their performance, how well they work with existing simulation algorithms, and how accurately they model certain aspects of the system.

1.2.1 Linear Rigid-body Models

Much work has been done in the modeling of rigid body impact sounds since the pioneering work of van den Doel et al. [138]. The basic approach is to compute vibra-

tion modes for a given rigid body shape, either analytically for simple shapes or by an eigenanalysis of a finite-element (FEM) system, and then excite these modes at run-time while treating them as independent harmonic oscillators. This builds upon earlier ideas of modal synthesis from the computer music community [2], where the technique is often used for synthesizing musical instruments rather than animation soundtracks. O'Brien et al. [103] demonstrated that this was a practical approach for synthesizing soundtracks for standard rigid body simulations. James et al. [67] improved the sound model by accurately computing how efficiently each mode radiates from the shape of the object, resulting in more accurate and interesting variety depending on the listening direction. Alternatively, a data-driven variation on the method is to measure vibration modes by exciting real, physical specimens via a scanning process [104, 117].

Much work has also been done in making modal models more efficient for real-time applications. Van den Doel et al. [139] proposed a run-time mode-culling technique that exploits human perceptual properties to reduce the number of modes that need to be integrated. Bonneel et al. [16] explored further performance improvements by synthesizing in the frequency domain rather than the time domain. Raghuvanshi et al. [112] propose the use of simpler mass-spring systems rather FEM to simplify the pre-process, and they utilize further run-time optimizations for large-scale applications.

Other enhancements to rigid body models include the incorporation of scraping and sliding sounds by modeling the roughness of the surface [140, 115]. Issues concerning the behavior of rigid body sound models in high-contact scenarios were addressed in [145]. The model can also be used to sonify fracturing bodies by treating each fractured piece as a rigid body, possibly approximated by existing models in a database [144]. Lastly, Chadwick et al. [25] enhance high-speed impact sounds by adding acceleration noise, a

phenomena that is not modeled by modal models alone.

1.2.2 Other Phenomenon

Beyond rigid bodies, physics-based methods for other systems have also been extensively researched. O'Brien et al. [102] proposed a method for sonfiying finite element deformable bodies by performing a surface integral at audio-rates. While they achieve plausible results for a variety of objects, the cost of audio-rate finite element simulation can be prohibitive for high-resolution models. Chadwick and James [24] synthesize sound for fire by analyzing combustion behavior in the underlying simulation. Two methods for sonifying fluids were proposed by Zheng and James [143] and Moss et al. [97], both based on the premise that most fluid sounds come from the vibrations of bubbles produced during splashing motions. They differ mainly in how they model the frequencies of the bubbles and how they handle the propagation of acoustic waves throughout the fluid domain. Dobashi et al. [41, 42] use physics-based sound textures, computed in an expensive pre-process with high-resolution fluid simulations, to synthesize sound in real time for aerodynamic and vortex sounds. Using some emperically derived relationships, they are able to modify the pre-computed results to be synchronized with a real-time animation. Another particularly well-studied phenomenon is thunder sounds from lightning. Ribner and Roy [116] proposed that acoustic emissions from lightning could be modeled as a superposition of many so-called *N-waves*. Glassner [50] builds upon their approach, but instead uses a different primitive called the *WM-wave* for improved accuracy and efficiency. Matsuyama et al. [90] extends Glassner's work by synchronizing it to their method for animating lightning. Fontana and Bresin [45] propose a physics-inspired sound model for the crumpling of an aluminum can. It is

based on an energy-conservation model driven by a Poisson process, and it produces interesting results for a relatively simple model.

1.3 Prior Work: Data-driven Methods

Data-driven methods are typically employed when the underlying phenomenon is too computationally expensive to simulate, or when sufficiently accurate physics-based models simply do not exist. This tends to be true for more organic and heterogenuous objects as well as aggregate phenomena. To get around such limitations, data-driven methods instead rely on actual recordings as the basis of their results. Most methods attempt to reproduce characteristics of the recording while still being synchronized to an input excitation, such as an animation. The disadvantage is a loss of generality, as the method will usually be limited by the richness and variety of the original recording. Furthermore, data-driven methods are usually not predictive and should not be relied on if physical accuracy and correctness are important to the application. Methods differ in how they analyze and synthesize the sound, which characteristics they consider to be important to reproduce, and how they synchronize to animation if they do so at all. In addition to the work described below, Strobl et al. [130] offers a survey of data-driven synthesis techniques, particularly focusing on *sound textures*.

1.3.1 Spectral Analysis and Resynthesis

Cook [35] presents various techniques for analyzing and resynthesizing recordings. Many are based on spectral methods explored extensively by the computer music community [124, 119]. These typically take a recorded signal, analyze the spectral content (using the Fourier transform for example), and then resynthesize a similar but different sound by modifying the spectral characteristics in some way. Cook presents an example system which synthesizes foot step sounds for walking over various types of ground. It starts by analyzing a recording of someone walking over, say, a bed of gravel. It then performs various analysis steps to extract individual step sounds and analyze their spectral content as well as other statistics. This results in a parametric model which can be used to synthesize plausible sounds for an input of foot step events.

Many other systems follow a similar pattern. Peltola et al. [105] does this for hand-clapping sounds, a phenomenon that is likely impractical to model and simulate with a physics-based model. They take a recording of clapping, analyze it to produce a parametric model, and synthesize new clapping soundtracks using various methods of exciting the model, such as a Poisson process. Events may also be generated by a character animation.

1.3.2 Other Methods

Picard et al. [107] uses a similar pipeline to retarget recorded sounds of rigid bodies, but the analysis and synthesis procedures are based on granular synthesis techniques [118].

They do use spectral methods to classify and delineate various categories of sound in the recording, such as impacts versus continuous rolling. The algorithm ultimately uses the recording by cutting it up into small samples and playing them back in a controlled, semi-randomized manner. The playback is synchronized to animation by analyzing rigid body simulation events and classifying them into analogous categories.

Dubnov et al. [43] propose the use of wavelet trees as a method for analysis and synthesis. The algorithm takes an input recording and performs a hierarchical wavelet transform, resulting in a wavelet tree. Then by randomly rearranging the nodes while respecting certain statistical characteristics of the original tree, the algorithm can produce novel sounds that still sound like the original. Using this method, they were able to re-synthesize many interesting phenomena, such as a traffic jam, a baby crying, and a car race. This synthesis process can also be done in a constrained manner such that it is synchronized to animation, and this is the approach presented by Cardle et al. [22]. They present various modes of control, such as a direct constraint specification user interface, and a semi-automated algorithm based on analyzing a given soundtrack-animation pair.

McDermott et al. [93] propose yet another approach to analysis and synthesis based on filter statistics. The method analyzes a given recording using a set of frequency subband filters and then computes various statistics within and between bands. In the synthesis process, it starts with random noise and iteratively applies the filter statistics to make them match the analysis. This produced convincing results for complex phenomena such as rain, wind, and ocean waves. It is unclear how this method would be synchronized to an animation, but some sort of constrained iterative optimization process may be the answer.

Lastly, *concatenative sound synthesis* (CSS) is another framework for various sound synthesis problems that has had good success in music and speech synthesis fields [121, 122, 65]. To quote Schwarz [121], "Concatenative data-driven sound synthesis methods use a large database of source sounds, segmented into heterogeneous units, and a unit selection algorithm that finds the units that match best the sound or musical phrase to be synthesised, called the target. The selection is performed according to the features of the units. These are characteristics extracted from the source sounds, e.g. pitch, or attributed to them, e.g. instrument class. The selected units are then transformed to fully match the target specification, and concatenated." Even though the problem domain is completely different, this is the approach taken for cloth sounds in Chapter 5.

1.3.3 Hybrid Methods

It would be inaccurate to view physics-based methods and data-driven methods as two disjoint categories. In reality, many techniques contain flavors of both extremes. The fire sound method of [24] is partially physics-based, but they found that there was insufficient variety and naturalness in the initial results. To enhance them, they utilize a texture synthesis method which adds detail based on recordings of real flames. The methods for aerodynamic and vortex sounds presented in [41, 42] can also be viewed as partially data-driven, except the data is from a physical simulation.

1.4 Contributions

The remaining chapters of this thesis present three novel contributions to the area of sound synthesis:

1. **Cubature Optimization for Thin Shell Sounds:** The first contribution comes in two chapters, as the key idea has more general applications beyond sound synthesis. Simulating highly deformable models is a challenging problem in any context, sound synthesis and otherwise. In order to achieve accurate results for complex objects, high resolution meshes must be employed, and these result in very high-dimensional systems that are expensive to time-step. One approach to making this more efficient is to perform model reduction, where the model's deformations are limited to a relatively small subspace. However, even with this reduction, internal forces must still be integrated over the full mesh. Cubature optimization addresses this issue by building an efficient approximation to these internal forces. Chapter 2 presents this method in the context of deformable body simulation in general.

   Chapter 3 applies cubature to the problem of physics-based thin shell sound synthesis. Thin shells, while they may not deform very much visually, are still nonlinear enough such that linear modal sound models cannot synthesize sufficiently realistic sounds. This is most easily observed by exciting a model with the same forces but at increasingly larger magnitudes. A linear sound model would simply produce the same waveform with some linear scale factor. In reality, thin shells, such as crash cymbals, sound very different depending on the magnitude of the impact. In order to capture this behavior, cubature is used to couple the vibration modes, resulting in much richer synthesis results.

2. **Compressing Modal Displacement Fields for Rigid-Body Sound Synthesis:** One of the major drawbacks of modal rigid body sound models is the large memory cost of using them at run-time. In order to determine how a given impact will excite the modes, the displacement field for each vibration mode must be evaluated at the point of impact. As vector fields, storing this information can take dozens of megabytes for typical models. We propose a method for compressing such fields using mesh simplification. Furthermore, for models which exhibit cylindrical symmetry, we show that further compression can be achieved by automatically detecting and exploiting this symmetry. This is the focus of Chapter 4.

3. **Motion-driven Concatenative Sound Synthesis of Cloth Sounds:** Cloth is a highly deformable material that has become an important topic in computer graphics. However, prior sound synthesis methods cannot practically produce plausible sounds synchronized to cloth animations. A physics-based method, such as the one presented by O'Brien et al. [102], would be impractically expensive for capturing all the subtle vibrational behaviors of cloth, assuming there even exists a mechanical model that models said behaviors. In Chapter 5, we present a data-driven method, building upon existing work from the speech and musical synthesis communities, which is able to efficiently synthesize plausible soundtracks synchronized to cloth animations for various materials.

CHAPTER 2

OPTIMIZING CUBATURE FOR EFFICIENT INTEGRATION OF SUBSPACE DEFORMATIONS

In order to achieve a physics-based sound model for a deformable body, we must first efficiently simulate the vibrational dynamics of the body. One approach is to reduce the dimensionality of the system by simulating it in a relatively small subspace of deformations. In this chapter, we propose an efficient scheme for evaluating nonlinear subspace forces (and Jacobians) associated with subspace deformations. The core problem we address is efficient integration of the subspace force density over the 3D spatial domain. Similar to Gaussian quadrature schemes that efficiently integrate functions that lie in particular polynomial subspaces, we propose cubature schemes (multi-dimensional quadrature) optimized for efficient integration of force densities associated with particular subspace deformations, particular materials, and particular geometric domains. We support generic subspace deformation kinematics, and nonlinear hyperelastic materials. For an r-dimensional deformation subspace with $O(r)$ cubature points, our method is able to evaluate subspace forces at $O(r^2)$ cost. We also describe composite cubature rules for runtime error estimation. Results are provided for various subspace deformation models, several hyperelastic materials (St.Venant-Kirchhoff, Mooney-Rivlin, Arruda-Boyce), and multimodal (graphics, haptics, sound) applications. We show dramatically better efficiency than traditional Monte Carlo integration. This work was originally published in An et al. [4].

2.1 Introduction

Recently, dimensional model reduction has gained attention due to the difficulty of simulating detailed physical models at high rates for multimodal (graphics, haptics, sound) applications. Such reduced-order methods construct a small, r-dimensional subspace that captures the salient features of a much larger, N-dimensional model. If $r \ll N$, simulating these reduced-order models entirely in the r-dimensional subspace holds the promise of superior runtime performance provided costs independent of N are achieved.

Unfortunately, efficient evaluation of internal forces for subspace deformation models has proven difficult for arbitrary geometry and nonlinear materials. In particular, the inability of many models to support more complex materials, such as biological materials for surgical simulation, is unfortunate. Current force evaluation methods either have costs dependent on N [78] or that scale poorly ($O(r^4)$) or are essentially restricted to particular materials (St. Venant-Kirchhoff) in practice [8], or are inaccurate.

In this paper, we present a general-purpose force evaluation method that applies to more general materials, subspace kinematics, and geometry, while delivering fast N-independent force evaluations at $O(r^2)$ cost (without exploiting sparsity). We achieve this scalability by performing a cubature optimization preprocess that enables fast runtime evaluation. Additionally, we provide evidence that our cubature schemes are computationally accurate and efficient, are resistant to over-fitting, and provide clear improvements over traditional Monte Carlo integration [5].

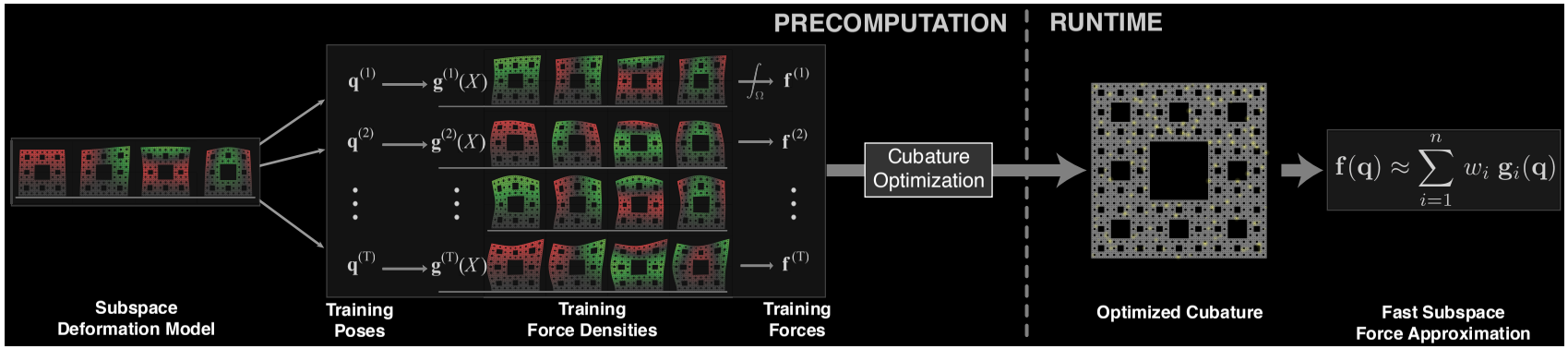
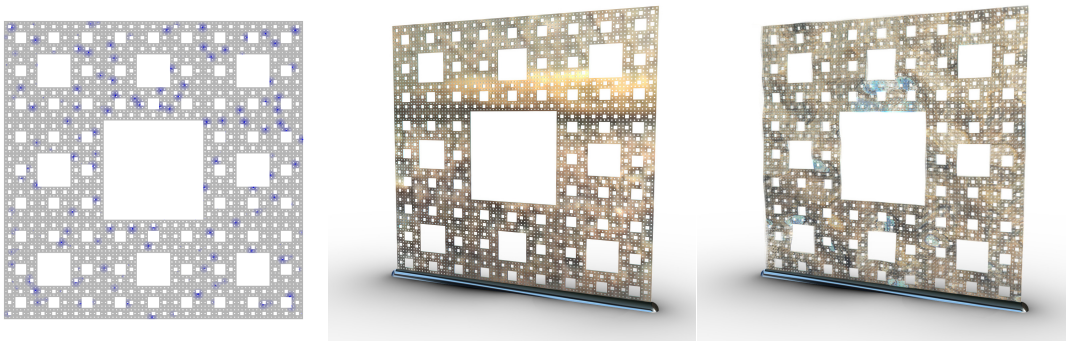


Figure 2.1: **Overview of Cubature Optimization:** *Given a subspace deformation model with N elements, we generate (sample or simulate) a set of training poses for input to the cubature optimization preprocess. The optimization procedure estimates $n \ll N$ cubature elements/points, and associated nonnegative weights, w_i, such that the the cubature approximation of $\mathbf{f}(\mathbf{q})$ well-approximates the training force/pose data. At runtime, the cubature scheme uses the force response of only the n cubature elements to provide a fast approximation to internal forces, therein accelerating integration of subspace deformation dynamics.*



OPTIMIZED CUBATURE REST POSE NONLINEAR VIBRATIONS

Figure 2.2: **Optimizing cubature for nonlinear modal sound:** *(Left) A 900-element cubature scheme optimized for integrating the nonlinear 200-mode subspace vibrations of (Middle) a complex Menger-inspired thin shell (one cube thick) modeled with 393216 tetrahedra, and a St.Venant-Kirchhoff material model of aluminum. (Right) Nonlinear shell vibrations (amplified* $4\times$ *for display). Optimized cubature permits explicit Newmark subspace integration of audio-rate (44.1 kHz) nonlinear sound simulations, with significant and audible nonlinear mode coupling effects, at greatly reduced costs: for a 5.0 second sound clip, nonlinear sound synthesis using optimized cubature and subspace integration (*$\Delta t = 1ms/88.2$*) required 3.5 single-core hours for subspace vibration (and radiation calculations), whereas sounds integrated using a parallelized implementation of subspace-projected unreduced forces [Krysl et al. 2001] required 4 days on 16 cores. The resulting sounds are virtually indistinguishable.*

Subspace Internal Forces: Our method can be applied to general subspace deformation models, but for concreteness of exposition we will focus on the case of dimensional model reduction for detailed finite element meshes [78, 8]. In a full FEM simulation with N degrees of freedom, the displacement vector would be of length N. However, in dimensional model reduction, the equations of motion have been projected into a linear, r-dimensional subspace of deformations. The reduced-order equations of motion describing a mesh deforming in subspace coordinates can be written as

$$\mathbf{M}\,\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q}) = \mathbf{f}_{\mathbf{ext}}, \tag{2.1}$$

where, $\mathbf{M} \in \mathbb{R}^{\mathbf{r}\times\mathbf{r}}$ is the (often constant) mass matrix, $\mathbf{q} \in \mathbb{R}^{r}$ is the generalized displacement vector of reduced coordinates, $\mathbf{f}(\mathbf{q}) \in \mathbb{R}^{\mathbf{r}}$ is the subspace internal restoring force, $\mathbf{f}_{\mathbf{ext}} \in \mathbb{R}^{\mathbf{r}}$ are external forces, and the overdot denotes differentiation.

Unfortunately, the subspace internal force term $\mathbf{f}(\mathbf{q})$ in (2.1) is responsible for the poor $O(rN)$ and $O(r^4)$ scalings of previous methods [78, 8], so its efficient evaluation is the focus of this paper. The subspace force can be formulated in terms of a potential energy function, $E(\mathbf{q}) : \mathbb{R}^r \rightarrow \mathbb{R}$, given by the domain integral,

$$E(\mathbf{q}) = \int_\Omega \Psi(X;\mathbf{q})\, d\Omega_X, \tag{2.2}$$

where $\Psi(X;\mathbf{q})$ is the nonnegative strain energy density at material point X of the undeformed material domain Ω [15]. The subspace internal force is then the gradient of this energy, and is given by the vector integral

$$\mathbf{f}(\mathbf{q}) = -\nabla_\mathbf{q} E(\mathbf{q}) = -\int_\Omega \nabla_\mathbf{q} \Psi(X;\mathbf{q})\, d\Omega_X = \int_\Omega \mathbf{g}(X;\mathbf{q})\, d\Omega_X, \tag{2.3}$$

where we denote the "reduced-force density" integrand by

$$\mathbf{g} = \mathbf{g}(X;\mathbf{q}) = -\nabla_\mathbf{q} \Psi(X;\mathbf{q}) \in \mathbb{R}^r. \tag{2.4}$$

Our approach is to approximate $\mathbf{f}(\mathbf{q})$ using an n-point cubature (multi-dimensional quadrature) scheme,

$$\mathbf{f}(\mathbf{q}) = \int_\Omega \mathbf{g}(X;\mathbf{q})\, d\Omega_X \approx \sum_{i=1}^{n} w_i\, \mathbf{g}(X_i;\mathbf{q}). \tag{2.5}$$

We precompute estimates of the n positive cubature weights (w_i), and n cubature points (X_i) by minimizing $\mathbf{f}(\mathbf{q})$ integration error over a training set of $(\mathbf{q},\mathbf{f}(\mathbf{q}))$ pairs. Our proposed preprocess is a greedy algorithm that incrementally selects cubature sample points, and estimates their nonnegative cubature weights. In our discrete implementation, each cubature point X_i corresponds to a linear tetrahedral element, since the force density Ψ is constant over each element. Runtime evaluation of subspace forces consists of evaluating only n deformed tetrahedra, and accummulating their $\mathbf{f}(\mathbf{q})$ contribution. An overview of our preprocess and runtime pipeline is shown in Figure 2.1.

Although no formal theory exists for cubature over nontrivial 3D domains, our empirical evidence indicates that cubature schemes can be optimized for efficient subspace force evaluation for (1) particular geometric domains, (2) particular materials, (3) particular deformation subspace kinematics and/or motion examples, and (4) greatly accelerated subspace force evaluation. See figure 2.2 for a preview of our results.

2.2 Other Related Work

For more than two decades, following the pioneering work of Terzopoulos, Barr, Witkin, and others, the mathematical foundations of Lagrangian dynamics have been employed in computer graphics to build dynamic physically based models of parametrized deformable shapes [133, 134, 142]. Monte Carlo methods were widely used to evaluate subspace force integrals (2.3); for example, Baraff and Witkin [5] mention that the gradient of the potential energy integral could be easily computed for relatively simple examples (such as a quadratically deforming block) using Monte Carlo integration: "For second-order polynomial deformations, a small number of sample points (on the order of fifty) yields adequate results." Unfortunately, we observe (Figure 2.8) that Monte Carlo is inefficient for more complex geometry, deformations, and materials.

Our approach is inspired by Gaussian quadrature and related schemes from classical 1-D numerical integration [58, 109], e.g., an n-point Gaussian quadrature scheme for a proper integral is

$$\int_{-1}^{1} g(X)dX = \sum_{i=1}^{n} w_i \, g(X_i), \tag{2.6}$$

where w_i are n (positive) weights, and X_i are n abscissae chosen as roots of a suitable

orthogonal polynomial. Surprisingly, with only n quadrature samples, Gaussian quadrature can evaluate integrals of polynomials of degree $2n-1$ *exactly*, so that each function sample effectively kills off a polynomial subspace of dimension two. If the function is very well approximated by a degree $2n-1$ polynomial, then the integral is also very well approximated. While Gaussian quadrature and related quadrature schemes (Gauss-Radau, Gauss-Lobatto, Radau, etc.) are widely used, variants for integrating higher-dimensional functions are restricted to tensor product domains and other simple parameterizations [58, 109]. In finite element analysis, integrals such as (2.3) are computed using related quadrature schemes but only for simple element shapes and low-order basis functions [10]. Exotically shaped elements have essentially avoided evaluation of (2.3) by using corotated linear models [76]. Sadly, no generalizations of Gaussian quadrature exist to nontrivial multi-dimensional domains (where it is called cubature), or to nonpolynomial function spaces relevant to subspace deformation forces.

Dimensional model reduction techniques use Galerkin projection onto a relatively low-dimensional linear subspace (spanned by the columns of the dense basis matrix, $\mathbf{U}$) to obtain a smaller reduced set of equations of motion, the unreduced internal forces, $\mathbf{F} = \mathbf{F}(\mathbf{Uq}) \in \mathbb{R}^N$ (evaluated in shape $\mathbf{Uq}$) are projected to yield reduced forces, $\mathbf{f} = \mathbf{U}^T\mathbf{F}(\mathbf{Uq}) \in \mathbb{R}^r$, equivalent to (2.3). Linear eigenmode coordinates are often used for subspace dynamics to resolve weak material nonlinearities in small-strain configurations [10] or mode-mode coupling [137]. For more nonlinear problems, Krysl et al. [78] formalized subspace integration for finite element models by employing a posteriori dimensional model reduction with "empirical eigenvectors" (or proper orthogonal decomposition (POD); principle component analysis) subspaces. The principle benefits are fewer ODEs to integrate, and smaller linear systems to solve (during implicit Newmark integration and Newton iterations). Unfortunately the speedup is fundamentally

limited by reduced force (and Jacobian) evaluation since they are based on "brute force" evaluation of $O(N)$ unreduced nodal force values ($\mathbf{F}$) in order to evaluate $\mathbf{f}$ via subspace projections ($\mathbf{U}^T\mathbf{F}$)–an $O(rN)$ cost. Related issues arise when computing forces and gradients for general-purpose multi-scale basis formulations, e.g., the basis-refinement formulation of CHARMS [53] (see also [21]) formalizes the (multi-resolution) scatter/gather integration steps of Galerkin subspace projection, but again invokes fine-scale evaluation of reduced force (and Jacobian) components for accurate evaluation of non-linear force response. Other schemes rely on coarsened discrete approximations for speed [39], albeit at the cost of geometric and/or material resolution.

Recently Barbič and James [8] observed that for the special case of St.Venant-Kirchhoff materials (large deformations, but linear stress-strain response), the reduced internal force (2.3) was in fact a vector of cubic polynomials in the reduced coordinates, $\mathbf{q} \in \mathbb{R}^r$. Subspace integrators can thus be generated for large-deformation reduced StVK models using arbitrary linear subspace bases, such as from PCA of training data, or linear and derivative modes using mass-PCA [8]. While extremely fast for small r, and suitable for real-time haptics [9], the cost complexity of the reduced force evaluation scales as $O(r^4)$ so that only models smaller than, e.g., $r=30$, offer significant speedups [7]. More general reduced kinematics would also be useful, but the model is limited to the linear basis superposition typical of POD methods. In contrast, our proposed approximation allows higher rank models due to its $O(r^2)$ force calculations for $O(r)$ cubature points.

For linear quasistatics, condensation and other precomputations can enable output-sensitive (subspace) evaluation of contact force/displacement responses [36, 69]. For linear elastodynamics, linear modal analysis allows efficient subspace force and dynamics models to be precomputed and diagonalized, thereby enabling $O(r)$ mode integration using IIR

filters [106, 70]. Linear modes have been warped to approximate large deformation kinematics [32].

Closely related to our approach, Key-Point Subspace Acceleration (KPSA) and caching have been proposed to accelerate posing of deformable characters [94]. The selection and use of key points is analogous to our selection of cubature points (or key elements). Unfortunately, KPSA does not solve the problem of estimating subspace forces and Jacobians associated with subspace deformations (although it was never intended to). For example, KPSA applied to subspace forces results in nonconservative force models with nonsymmetric Jacobians due to KPSA's use of least-squares estimation. Instead, it is more natural here to formulate reduced forces in terms of subspace derivatives of strain energy using the solid foundations of numerical integration.

Articulated subspace deformation models are commonplace in character animation, and can be used with cubature optimization. Pose space deformation interpolates shape correction coefficients as a function of pose [81]. For physics-based models [77], cubature could assist with estimating pose-specific subspace corrections potentially avoiding high-dimensional interpolation and training difficulties.

An alternative to runtime integration of $\mathbf{g}$ over Ω is to precompute a fast model of $\mathbf{f}(\mathbf{q})$ using various system modeling techniques, such as data-driven interpolation using radial basis functions [101] or neural networks [54]. Other approaches tabulate forces indirectly using compressed motion libraries for runtime playback, but simulation is done in an unreduced setting [68]. In such approaches, challenges include guaranteeing adequate data and model training, supporting high subspace dimensionality, avoiding over-fitting, and ensuring energy conservation, passivity, and stability. To a large extent,

cubature avoids problems associated with approximating $\mathbf{f}(q)$ (or dynamics) by estimating and storing just $2n$ cubature values (n sample indices, and n nonnegative weights), thereby exploiting the redundant spatial structure of subspace deformation, and the energy integrand's functional structure.

2.3 Subspace Deformation Model

Subspace Kinematics: Given the time-dependent parameters of an r-dimensional subspace deformation, $\mathbf{q} = \mathbf{q}(t) \in \mathbb{R}^r$, the subspace deformation is specified pointwise by the deformation operator, $\varphi = \varphi(X;\mathbf{q})$,

$$x = \varphi(X;\mathbf{q}) \in \mathbb{R}^3, \tag{2.7}$$

where $x \in \mathbb{R}^3$ is the deformed image of the undeformed material point, $X \in \mathbb{R}^3$. Computationally, we assume that the cost of evaluating a deformed point using (2.7) is $O(r)$ flops. The partial derivatives of φ are important kinematic quantities: (1) the *deformation gradient*,

$$F = F(X,\mathbf{q}) = \frac{\partial x}{\partial X} \in \mathbb{R}^{3\times 3} \tag{2.8}$$

and (2) the *displacement sensitivity matrix*,

$$U = U(X,\mathbf{q}) = \frac{\partial x}{\partial \mathbf{q}} \in \mathbb{R}^{3\times r}. \tag{2.9}$$

Materials are defined via the strain energy density, $\Psi(X;\mathbf{q})$, which are used to evaluate the deformation potential energy and subspace force integrals (2.2-2.3). Materials used in this paper are given in Appendix 2.6.3.

Discrete Setting: To support reduced-order model construction, we will employ an underlying discrete model. Without loss of generality, our implementation uses tetrahedral finite element models with linear shape functions. We will refer to nodal positions as a vector of position quantities: given N_v nodal vertices, let the undeformed material positions be $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{N_v})^T \in \mathbb{R}^{3N_v}$, and the deformed positions be $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_v})^T \in \mathbb{R}^{3N_v}$, such that $\mathbf{x}_i = \varphi(\mathbf{X}_i; \mathbf{q})$. An important displacement-like matrix quantity is the gradient of the position with respect to $\mathbf{q}$,

$$\frac{\partial \mathbf{x}}{\partial \mathbf{q}} \equiv \mathbf{U} = \mathbf{U}(\mathbf{q}) \in \mathbb{R}^{3N_v \times r}. \tag{2.10}$$

For the important case of linear shape models, $\mathbf{U}$ is a constant matrix of displacement modes, which has the important consequence of making the mass matrix $\mathbf{M}$ constant.

2.4 Discrete Cubature

We use optimization to estimate an n-point cubature scheme (2.5) that approximates the reduced force $\mathbf{f}(\mathbf{q})$ integral in (2.3). It follows that our approximation of the gradient of the reduced force vector, or the stiffness matrix, is

$$\mathbf{K}(\mathbf{q}) \approx \sum_{i=1}^{n} w_i \frac{\partial \mathbf{g}(X_i; \mathbf{q})}{\partial \mathbf{q}^T}. \tag{2.11}$$

Observe that by choosing nonnegative weights, $w_i \geq 0$, we are guaranteed that $\mathbf{K}$ inherits the same semi-definiteness properties of the integrand.

Element-based Cubature Schemes: Cubature optimization would suggest considering continuous positions, $X \in \Omega$, and thus a continuous-valued, constrained optimization

problem. However, in reduced-order modeling we may only have access to discrete representations of the integrand. Therefore, we consider discrete optimization schemes, wherein the candidate cubature "points" (and hence $\mathbf{g}$) are chosen from a finite set.

In our implementation, we use linear tetrahedral finite elements with constant deformation gradients. The tetrahedrons are features which can be seen as generalized cubature "points." The reduced-force integrand $\mathbf{g}(X_i;\mathbf{q})$ of any point $X_i \in \Omega$ is equal to the response of the containing element.

Complexity of Cubature Evaluation: The cost of evaluating internal forces (2.5) using an n-point cubature scheme is $O(rn)$, since each cubature point's contribution can be accumulated in $O(r)$ operations (assuming dense matrices and global deformation support). For example, a tetrahedron used in a cubature scheme can be deformed in $O(r)$ operations using (2.7), the resulting 12-vector of vertex forces can be computed at $O(1)$ cost, and these forces can be projected into subspace force contributions using a 12-by-r matrix-vector multiply at $O(r)$ cost. Evaluating the stiffness matrix approximation (2.11) involves $O(r^2n)$ cost (assuming dense matrices). For example, a 12-by-12 tetrahedral stiffness matrix, $\mathbf{K}_e$, can be computed using (2.7) at $O(r)$ cost, however computing its contribution to the r-by-r subspace stiffness matrix $\mathbf{K}(q)$ involves a subspace projection of the form, $\mathbf{U}_e^T\mathbf{K}_e\mathbf{U}_e$, which incurs an $O(r^2)$ cost.

Assuming $n \propto r$ (as we observe in practice (§2.7)), we therefore obtain $O(r^2)$ cost for force evaluation and $O(r^3)$ cost for stiffness evaluation. However, we show in §2.6.2 that a fast $O(r^2)$ stiffness matrix-vector product is possible if the stiffness matrix need not be formed explicitly. Finally, while complexity analysis can show costs independent of N, an important result is that subspace computations are still fast in practice (see

Figure 2.3).

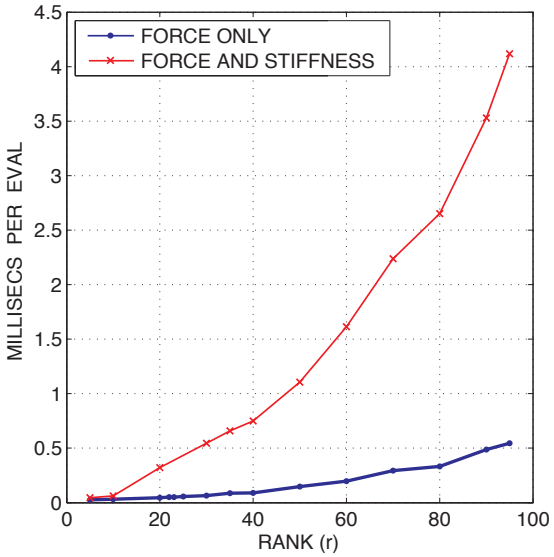


Figure 2.3: **Complexity of force and stiffness evaluation:** *Assuming $n = O(r)$ cubature samples, internal force evaluation costs $O(r^2)$ flops, whereas dense stiffness matrix evaluation is $O(r^3)$ (although a fast matrix-vector multiply exists (§2.6.2)). These timings were done using $n = r$ cubature schemes.*

2.5 Optimizing Cubature

We model the cubature optimization problem as a discrete subset selection problem, where we attempt to select cubature points/elements that, when weights are optimized using nonnegative least squares (NNLS), will tend to minimize fitting error of (2.3). The optimization estimates cubature quality via the error of the subspace force estimate using T training data samples, $\{(\mathbf{f}^{(t)}, \mathbf{q}^{(t)})\}_{t=1\ldots T}$ where we compute $\mathbf{f}^{(t)} = \mathbf{f}(\mathbf{q}^{(t)})$ for the shape $\mathbf{q}^{(t)}$ using standard methods, e.g., subspace projection of unreduced forces [78]. We now describe the procedures for estimating weights (§2.5.1), greedy cubature optimization (§2.5.2), training data generation (§2.5.3), and cubature validation (§2.5.5).

2.5.1 Estimating Nonnegative Cubature Weights

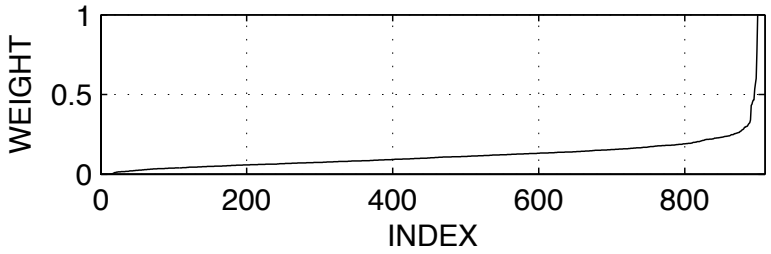
Given a set $\mathcal{S}$ of n cubature points, we estimate cubature weights $\mathbf{w} \in \mathbb{R}^n_+$ (that multiply the integrand samples, $\mathbf{g}_i^{(t)} = \mathbf{g}(X_{S_i}; \mathbf{q}^{(t)})$), by minimizing the error in predicting (2.3) over the training data's T reduced force values, $\mathbf{f}^{(t)}$, $t = 1 \dots T$. To avoid over-fitting and preserve the spectral properties of stiffness matrices, we estimate nonnegative weights using nonnegative least squares (NNLS) by solving:

$$\begin{bmatrix} \frac{\mathbf{g}_1^{(1)}}{\|\mathbf{f}^{(1)}\|} & \cdots & \frac{\mathbf{g}_i^{(1)}}{\|\mathbf{f}^{(1)}\|} & \cdots & \frac{\mathbf{g}_n^{(1)}}{\|\mathbf{f}^{(1)}\|} \\ \vdots & & \vdots & & \vdots \\ \frac{\mathbf{g}_1^{(t)}}{\|\mathbf{f}^{(t)}\|} & \cdots & \frac{\mathbf{g}_i^{(t)}}{\|\mathbf{f}^{(t)}\|} & \cdots & \frac{\mathbf{g}_n^{(t)}}{\|\mathbf{f}^{(t)}\|} \\ \vdots & & \vdots & & \vdots \\ \frac{\mathbf{g}_1^{(T)}}{\|\mathbf{f}^{(T)}\|} & \cdots & \frac{\mathbf{g}_i^{(T)}}{\|\mathbf{f}^{(T)}\|} & \cdots & \frac{\mathbf{g}_n^{(T)}}{\|\mathbf{f}^{(T)}\|} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_i \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{f}^{(1)}}{\|\mathbf{f}^{(1)}\|} \\ \vdots \\ \frac{\mathbf{f}^{(t)}}{\|\mathbf{f}^{(t)}\|} \\ \vdots \\ \frac{\mathbf{f}^{(T)}}{\|\mathbf{f}^{(T)}\|} \end{bmatrix} \Leftrightarrow \mathbf{A}\mathbf{w} = \mathbf{b}, \tag{2.12}$$

subject to nonnegativity constraints, $\mathbf{w} \geq \mathbf{0}$. Here $\mathbf{A}$ is a dense rT-by-n matrix, and $\mathbf{b}$ is an rT-vector. Each r-vector in row t of $\mathbf{A}$ and $\mathbf{b}$ is scaled by $\|\mathbf{f}^{(t)}\|^{-1}$ in order to *minimize relative error instead of absolute error*. Larger absolute errors will be tolerated for larger training forces, so a small number of very large samples will not distort the fitting process.

Such problems can be solved efficiently using available NNLS implementations [80]. In practice, we can estimate nonnegative cubature weights with one NNLS call in less than a minute for a relatively complex model with $T = 1000$ poses, $r = 100$ dimensions, and $n = 200$ samples. However, for challenging examples, such as the "Menger shell" (Figure 2.2), NNLS calls can be expensive: 36 minutes for $T = 1000$, $r = 200$, and $n = 800$. Given that solve times scale roughly as $O(rTn^2)$, we will address the superlinear scaling of NNLS with n and r later in §2.5.4.

Example sorted nonnegative weights (for "Menger shell") are:



Error estimator, ε: In subsequent optimizations, we choose to minimize RMS relative L2-norm error over all samples, as described by the following error metric:

$$\varepsilon = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\frac{\|\tilde{\mathbf{f}}^{(t)} - \mathbf{f}^{(t)}\|^2}{\|\mathbf{f}^{(t)}\|^2}}. \tag{2.13}$$

All following training and validation convergence plots were done using this metric. Given the scaling by $\|\mathbf{f}^{(t)}\|^{-1}$, this is equivalent to the relative residual error $\|\mathbf{r}\|/\|\mathbf{b}\|$ optimized by the NNLS problem and the greedy algorithm that follows.

2.5.2 Greedy Estimation of Cubature Points

We propose a simple, iterative, greedy, subset-selection algorithm[1]. At every iteration, we add a key element e such that $\mathbf{g}_e$ is the most positively parallel to the current NNLS residual. This will reduce the size of the residual the most. Then, we update the residual and iterate again. The algorithm is as follows:

Here $\mathcal{S}$ is the set of key elements, $\mathbf{r}$ is the current NNLS residual, $\mathbf{w}$ is the vector of cubature weights, and $\mathbf{A}_{\mathcal{S}}$ is the $\mathbf{A}$ matrix with columns corresponding to elements in

[1] Our greedy algorithm is analogous to one proposed for multipole source placement [67], although here the data is real-valued, and NNLS solves are used at each iteration.

Algorithm 1: Greedy cubature optimization algorithm

```
begin
    S ← ∅
    r ← b
    while ‖r‖/‖b‖ > TOL do
        C ← SelectCandidatePoints(S)
        e ← arg max_{e∈C} (g_e^T r)/(‖g_e‖‖r‖)
        S ← S ∪ {e}
        w ← NNLS(A_S, b)
        r ← b − A_S w
    return (S, w)
end
```

$\mathcal{S}$. TOL is the user-specified error tolerance, and once the algorithm achieves an error below the tolerance, it terminates.

Lazy Evaluation: The sub-function SELECTCANDIDATEPOINTS must choose the remaining elements/points most likely to reduce the residual error. The most thorough implementation is to return all remaining elements not already in $\mathcal{S}$. However, for high-resolution meshes, storing or calculating the $\mathbf{g}_e$ vectors for all elements is impractical. Instead, we pick a random subset of the remaining elements and calculate the $\mathbf{g}_e$ vectors needed for the arg max step. The full $\mathbf{A}$ matrix is never actually computed or stored, so the algorithm has modest memory requirements, even for meshes with hundreds of thousands of elements. These column computations are also trivial to parallelize.

One adjustable parameter of the algorithm is the number of random candidates, $|\mathcal{C}|$, to consider at each iteration. We found that for our examples, increasing the value of $|\mathcal{C}|$ beyond 1000 elements does not significantly improve the quality of the cubature after 10% error is reached. Figure 2.4 demonstrates that considering only 100 candidates per

iteration is about the same as considering 10000, suggesting that optimal selection at any greedy step is not critical.

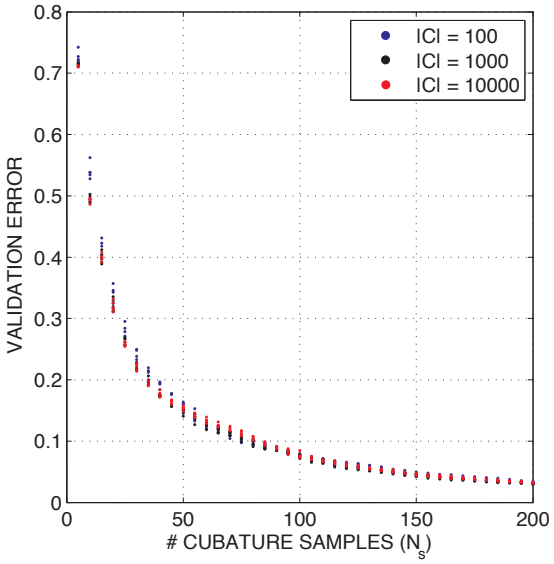


Figure 2.4: **Comparison of different values of $|\mathcal{C}|$:** *Below 10% error, the quality of the cubature does not degrade significantly when considering less candidates per iteration. Thus, the greedy cubature optimization algorithm is practical for high-resolution meshes. The bridge model of 29,800 elements was used for this plot.*

2.5.3 Training Data Generation

Various methods may be used to collect cubature training samples:

- A simple approach is to use pre-simulated motion for reanalysis [78] of non-interactive simulation conditions, possibly with forcing variations, e.g., in gravity, wind, or inflation conditions of a balloon. The **q** values produced during the simulation can then be recorded to disk and sampled for training and validation.

- Manual interaction (or user sketches) with a reduced simulation using fully projected forces can be used, but can be slow for large meshes. To work around this, manual input can be recorded and then simulated off-line [8].
- For a linear modal analysis basis used for small-strain subspace integration [10], training data can be generated automatically by randomly sampling a Gaussian distribution for each mode (component of $\mathbf{q}$), with standard deviations proportional to the inverse of the mode's frequency.

Once the $\mathbf{q}^{(t)}$ values are known for the poses, we can lazily compute the columns of $\mathbf{A}$ for training, as needed.

2.5.4 Optimization Complexity Analysis

Scaling with N: The algorithm cost and memory requirements are linear in the size of discrete model, which is mandatory for reduced-order modeling. This N-dependence arises from the computation of T training force values, $\mathbf{f}^{(t)}$. The "Menger shell" model illustrates that our method can support large tetrahedral models.

Scaling with r: For higher rank models, one may need hundreds of cubature points to achieve low error. For such problems, the optimization process can become impractical. Informally, NNLS exhibits $O(rTn^2)$ complexity, so that high r and n values can be slow. To limit this bottleneck, we use *subset training,* wherein NNLS regression is done on a small subset of T_s training samples. Before the NNLS call, we randomly choose this

subset, perform NNLS only on the subset to update the residual, and then on the next iteration we greedily select the point most fit with respect to the residual. Although the residual is inaccurate, each iteration is much faster. However, one can occasionally do NNLS on the full training set to check the actual error and output a final cubature rule. In our experiments, we used $T_s = 10$ and do a full regression every $O(r)$ iterations to obtain cubature schemes that appear to achieve training errors within 5% of comprehensively trained ($T_s = T$) rules. For high-rank models, such as $r = 200$ for the detailed Menger shell, optimization times became hours instead of days.

Complexity Summary: The optimization bottleneck is repeated calls to the NNLS routine. Characterizing the cost of NNLS as $O(rTn^2)$, then since every iteration calls NNLS with one more cubature sample the total complexity is $O(rTn^3)$. If only a subset of the training set is used each iteration the cost is $O(rT_sn^3)$.

2.5.5 Cubature Validation

In some ways, the requirement of training data is a weakness of cubature optimization. However, a strength of cubature optimization, perhaps due to the fact that the method only learns cubature points and nonnegative weights, is that it resists overfitting. To assess the quality of a given cubature rule, we evaluate it on a validation set, $\{(\mathbf{f}^{(v)}, \mathbf{q}^{(v)})\}_{v=1\ldots V}$, obtained in ways similar to training set generation (§2.5.3). To estimate validation error, we use the training error metric (2.13) on the validation set. Our experiments show that cubature optimization is surprisingly resistant to over fitting: for all examples, the training and validation convergence plots are nearly identical. The

two representative plots are shown for the rope bridge example in Figure 2.5.

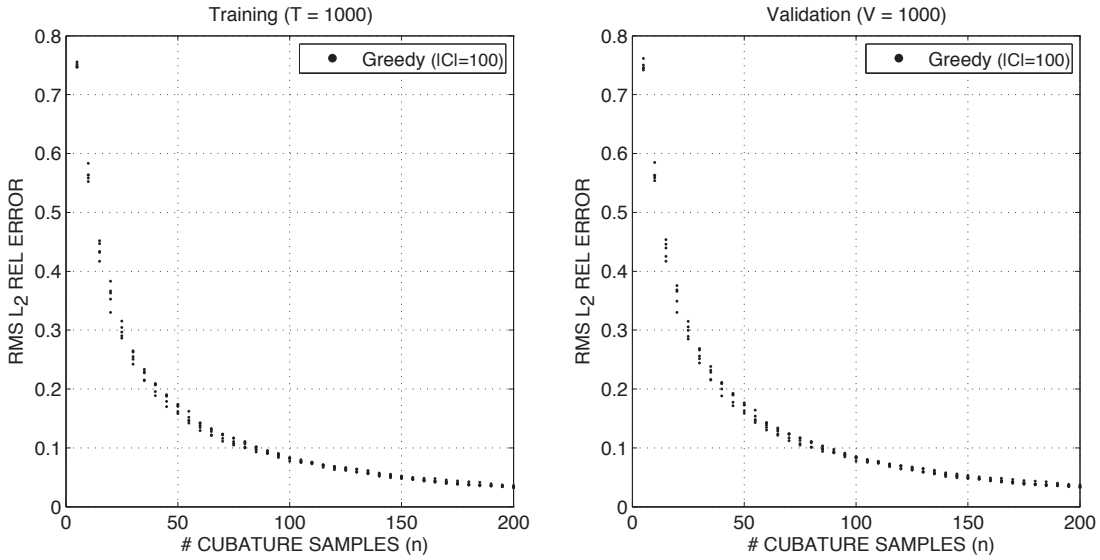


Figure 2.5: **Training and validation convergence plots** *for the rope bridge example ($r=100$) illustrate characteristic resistance of cubature optimization to over-fitting. Five cubatures were optimized for each n value.*

2.6 Implementation Details

2.6.1 Fast Integrand Evaluation

For a constant displacement sensitivity matrix $\mathbf{U}$, equation (2.5) can be accelerated using level-2 BLAS matrix-vector multiplies. Reshaping the deformation gradient for element e as a 9-vector, it can be calculated as $\mathbf{F_e} = \mathbf{E_e q} + \mathbf{I}$, where $\mathbf{E_e} \in \mathbb{R}^{9\times r}$ can be derived and pre-computed using $\mathbf{U}$ (c.f. [7]). Furthermore, all deformation gradients for cubature elements can be calculated in one matrix-vector multiply by stacking all $\mathbf{E_e}$ matrices into a single matrix $\mathbf{E} \in \mathbb{R}^{9n\times r}$. All energy density gradients with respect to the deformation

Model	Material	N_{tet}	Basis	Rank (r)	n	**f** Eval	**K** Eval
Bridge*	St. Venant-Kirchhoff	29,800	LMA	100	100	0.48 ms	3.0 ms
Balloon*	Mooney-Rivlin	118,208	PCA	30	90	0.15 ms	0.88 ms
Menger*	St. Venant-Kirchhoff	393,216	LMA	200	900	9.7 ms	75 ms
Haptic†	St. Venant-Kirchhoff	21,376	LMA	10	32	0.02 ms	0.19 ms
Haptic†	Mooney-Rivlin	21,376	LMA	10	12	0.01 ms	0.07 ms

Model	Material	Solve	Timesteps/sec	Optimization	Error
Bridge*	St. Venant-Kirchhoff	0.29 ms	265 (implicit)	47 secs	8%
Balloon*	Mooney-Rivlin	0.03 ms	943 (implicit)	34 secs	1.5%
Menger*	St. Venant-Kirchhoff	2.2 ms	101 (explicit)	1.8 hours	6.4%
Haptic†	St. Venant-Kirchhoff	0.008 ms	4230 (implicit)	10 secs	3%
Haptic†	Mooney-Rivlin	0.008 ms	8750 (implicit)	5 secs	3%

Table 2.1: **Model statistics** *including number of tetrahedra (N_{tet}); rank (r) of linear deformation basis* **U***; number of cubature points/elements (n); time to evaluate* both $\mathbf{f}(\mathbf{q})$ *and* $\mathbf{K}(\mathbf{q})$*; time to solve an r-by-r dense symmetric positive-definite linear system using LAPACK for implicit Newmark subspace integration; time-stepping rate of Newmark subspace integration used in demos, for either explicit or semi-implicit (one Newton-Raphson iteration) schemes; time for the cubature optimization preprocess (using 4-16 Xeon cores); relative training error* ε *of the resulting cubature scheme. With the exception of cubature optimization, the * timing experiments were done on a 2.4GHz Intel Core2, and the † experiments on a 3.0GHz Intel Xeon. The Intel Math Kernel Library was used for BLAS operations (dual-core enabled). Cubature training was done using the Greedy algorithm with* $|\mathcal{C}| = 100$ *(except for the balloon and haptic examples which used* $|\mathcal{C}| = 1000$*) and subset training (*$T_s = 10$*, full NNLS solves every* $r/2$ *iterations; except haptic examples use comprehensive training,* $T_s = T$*).*

gradient, $\tilde{\mathbf{g}}_i$, can then be evaluated for each sample element (invoking the constitutive model) in an $O(n)$ loop. Finally, the subspace projection and summation can be evaluated as another matrix-vector multiply, $\mathbf{f} = \mathbf{H}^T[\tilde{\mathbf{g}}_1 \dots \tilde{\mathbf{g}}_n]$, where the matrix $\mathbf{H} \in \mathbb{R}^{9n \times r}$ can be pre-computed using $\mathbf{U}$ and the cubature weights w_i. Similar optimizations exist for Jacobian matrix evaluation. In our implementation, this level-2 BLAS optimization provided nearly a two-fold speedup in force evaluation.

2.6.2 $O(r^2)$ Dense Stiffness Matrix-Vector Products

Fast matrix-vector product evaluation with the stiffness matrix, $\mathbf{K} \in \mathbb{R}^{r\times r}$, is often required for stiffness proportional Rayleigh damping, or implicit integrators. Unfortunately, for dense $\mathbf{U}$, the stiffness matrix is dense, and therefore evaluating each cubature sample's contribution to all entries of $\mathbf{K}$ involves $O(r^2)$ work, so that the total cost of forming the stiffness matrix is $O(r^2 n)$. Fortunately, many iterative Krylov-Newton solvers only require evaluation of matrix-vector products. Consequently we can exploit the fact that each cubature sample only contributes a low constant-rank update to $\mathbf{K}$, so that matrix-vector products, $\mathbf{Kv}$, can be constructed in $O(rn)$ (or $O(r^2)$ for $n = O(r)$). The $\mathbf{Kv}$ matrix-vector product can be written

$$\mathbf{Kv} = \sum_{e\in\mathbb{S}} w_e \frac{\partial \mathbf{g}(e;\mathbf{q})}{\partial \mathbf{q}^T}\mathbf{v} = \sum_{e\in\mathbb{S}} w_e \, \mathbf{U}_e^T \, \mathbf{K}_e \, \mathbf{U}_e \mathbf{v} \tag{2.14}$$

where $\mathbf{U}_e \in \mathbb{R}^{12\times r}$ is the displacement matrix for tetrahedral element e, and $\mathbf{K}_e \in \mathbb{R}^{12\times 12}$ is the element's stiffness matrix. The product $\mathbf{K}_e\mathbf{U}_e$ takes $O(r)$ flops to compute, and can be precomputed and cached. For the $\mathbf{Kv}$ product, we first compute $\mathbf{t} = (\mathbf{K}_e\mathbf{U}_e)\mathbf{v}$, and then $\mathbf{U}_e^T\mathbf{t}$, both of which take $O(r)$ flops per cubature sample.

2.6.3 Material Strain Energy Densities, Ψ

For our examples, we used the following strain energy densities, Ψ [15]. The St. Venant-Kirchhoff model is

$$\Psi(\mathbf{E}) = \frac{1}{2}\lambda tr(\mathbf{E})^2 + \mu \mathbf{E} : \mathbf{E}, \tag{2.15}$$

where λ and μ are the Lamé constants, and μ corresponds to the shear modulus. The Arruda-Boyce model uses

$$\Psi(\mathbf{E}) = \mu \sum_{i=1}^{5} \frac{C_i}{N^{i-1}} (I_C^i - 3^i), \tag{2.16}$$

where I_C is the first deviatoric strain invariant, μ is the initial shear modulus, and N is the number of rigid links for the model [83]. The Mooney-Rivlin constitutive model uses

$$\Psi(\mathbf{E}) = \mu_{10}(I_C - 3) + \frac{1}{2}\mu_{01}(I_C^2 - II_C - 6), \tag{2.17}$$

where II_C is the second deviatoric strain invariant, and μ_{10} and μ_{01} are material constants. The last two models do not enforce incompressibility, so we add a penalty term $K\log(J^2)$ to approximately conserve volume, where K is the bulk modulus, and $J = det(F)$.

2.7 Results

We now provide numerical results and analysis; please see our accompanying video for animation results (http://www.cs.cornell.edu/Projects/Sound/cubature/). Model statistics and algorithm timings are provided in Table 2.1. We note that graphical renderings were done using unoptimized implementations of ambient occlusion [59], and offline renderings.

Comparison to Gaussian quadrature: Although we do not target 1-dimensional or tensor-product integration applications, out of curiosity, we compared our Greedy-NNLS cubatures to Gaussian quadrature on the $[-1,1]$ interval. To do so, we randomly

generated suitably normalized degree-n polynomials using Chebyshev basis functions, then trained our cubature using $N = 1000$ uniformly distributed candidate abscissae. We observe that while an n-point Gaussian quadrature rule can exactly integrate a degree $2n-1$ polynomial, on average our n-point Greedy-NNLS scheme can integrate only an n-degree polynomial for 0.5% training error. Results are shown in Figure 2.6.

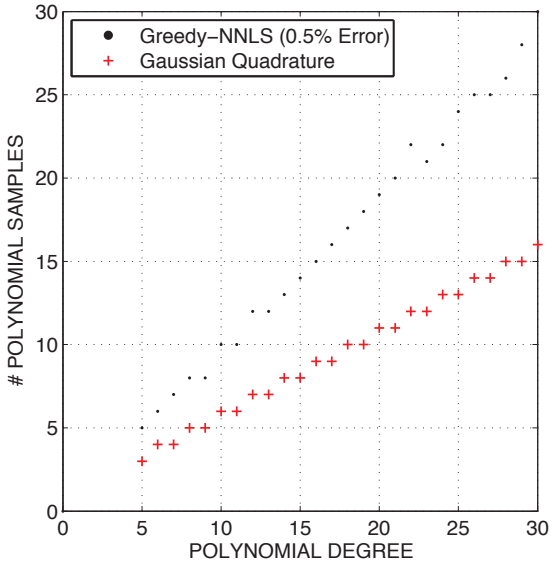


Figure 2.6: **Comparison to Gaussian quadrature:** *In the 1-D case, our greedily-optimized quadrature rules require about twice as many samples as the corresponding Gaussian quadrature rules to achieve 0.5% error.*

Scaling with rank for given error tolerances: Analogous to the 1-dimensional comparison, we also investigated how n scales with the rank of the reduced model. For a given reduced model of rank r, how many cubature samples n are necessary to achieve a given error tolerance? Figure 2.7 shows that the greedy algorithm can produce cubature rules that satisfy the error tolerance with $n = O(r)$ sample elements. The constant factor varies depending on the error tolerance and the geometry of the mesh. As expected, more cubature samples are needed to meet lower error tolerances.

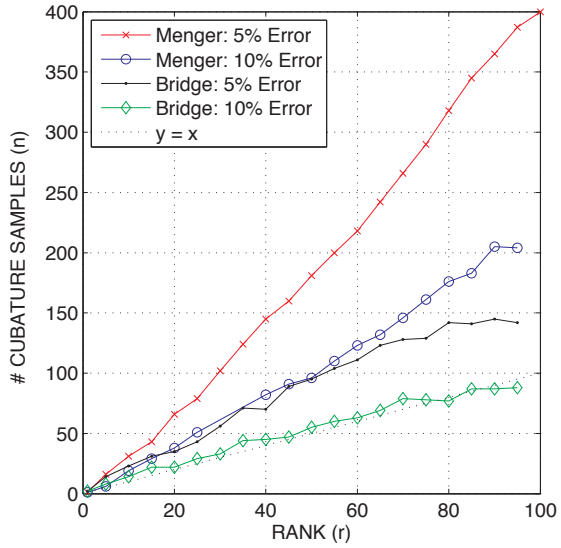


Figure 2.7: **Scaling with rank for given error tolerances:** *In practice, $n = O(r)$ cubature samples are sufficient to achieve a given error tolerance, but the constant factor depends on the example and the desired error tolerance.*

Cubature error analysis and comparisons: Figure 2.8 provides error convergence plots as a function of the number of cubature samples/elements used. The results indicate that our Greedy-NNLS cubature schemes tend to perform well in practice, especially for low error (ε) and high rank (r) situations, and is dramatically more efficient than Monte Carlo integration.

Error Estimation: Classical quadrature schemes often provide error estimators that can be used with little additional computation during evaluation. For example, a Gauss-Kronrod pair consists of an n-point Gaussian rule and a $(2n+1)$-point Kronrod rule that reuses all of the points from the Gaussian rule [56]. The Kronrod rule is used as the integral approximation, while the difference between the two rule values is an error estimate.

Analogously, given $2n$ cubature elements, we run NNLS on the first n elements to com-

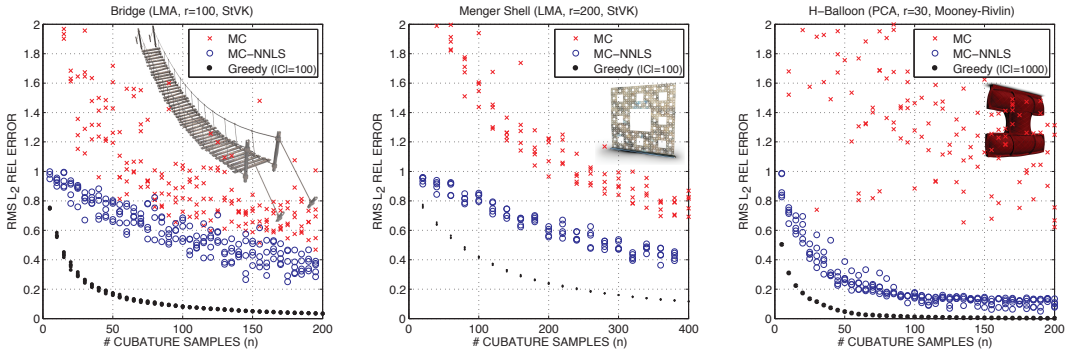


Figure 2.8: **Cubature Convergence Analysis:** *Plots of reduced-force training error, ε, as a function of the number of key elements, n. Results are shown for the uniformly weighted Monte Carlo scheme (MC), MC-sampled positions but NNLS-estimated weights (MC-NNLS), and our greedy approach (Greedy). We used $T = 1000$ training samples for the bridge and Menger shell, and $T = 50$ (from the full inflation simulation) for the balloon. For these plots, we do a full NNLS regression per-iteration, so $T_s = T$ (no subset training), in order to get accurate error measurements, however, subset training (§2.5.4) accelerates optimization, e.g., of our $n = 900$ Menger cubature scheme ($\varepsilon = 6.4\%$).*

pute the nonnegative weights of a coarser cubature rule C. The relative error estimate is calculated as $\|\mathbf{f}_C - \mathbf{f}\|/\|\mathbf{f}\|$ where $\mathbf{f}_C$ is the force estimated by the coarser rule C. Like a Gauss-Kronrod pair, this requires no additional evaluations of the integrand. Figure 2.9 illustrates estimator performance.

Comparison to reduced St.Venant-Kirchhoff: In Figure 2.10 we compare a cubature-based reduced-force evaluation to an optimized reduced St.Venant-Kirchhoff (StVK) model where reduced forces are exactly represented by an r-vector of polynomials cubic in the components of $\mathbf{q}$ [8]. Unlike our $O(r^2)$ approximation, exact evaluation of StVK reduced forces requires $O(r^4)$ operations, and can be prohibitive for larger r values. For a fair timing comparison, we use the optimized level-3 BLAS implementation of the authors of [8], and the level-2 BLAS implementation of our reduced-force evaluation (§2.6.2).

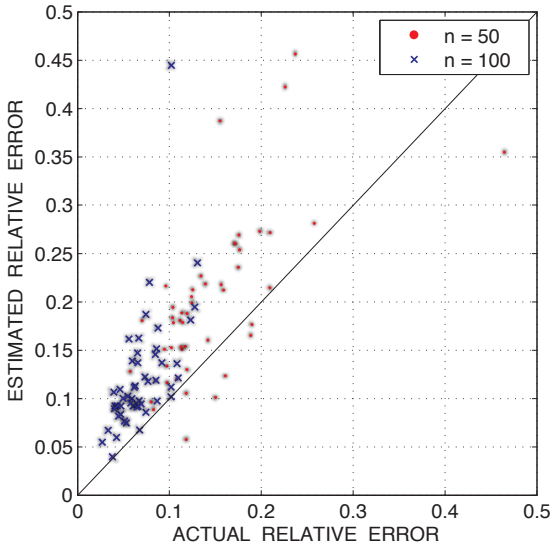


Figure 2.9: **Error Estimator** *results for two cubature pairs: $n=50$ (& 25) and $n=100$ (& 50) samples (rope-bridge example with $r=100$). Each point in the plot represents the validation error of a random deformable pose. Here the $n=100$ error estimate appears conservative, since its estimate is higher than the actual error, i.e., points are above the "$y=x$ line."*

Rope bridge: To illustrate that cubature can be optimized for structures with complicated topologies, we considered a jungle-like rope bridge (see Figure 2.11). This polygon soup model was discretized into 29,800 tetrahedra using a voxel embedding approach [8], and then approximated using an StVK model with tuned parameters. Because their dynamics looked reasonable, we used a linear modal basis shape model, and relied on the StVK nonlinearity to avoid large-deformation distortion.

Reanalysis: Another method of generating a basis is to perform proper orthogonal decomposition (POD), or principle component analysis (PCA), on full simulation data [78]. For our example, we simulate a balloon being inflated by constant air pressure forces (see Figure 2.12). Our reduced model can then re-create the dynamics of inflation at

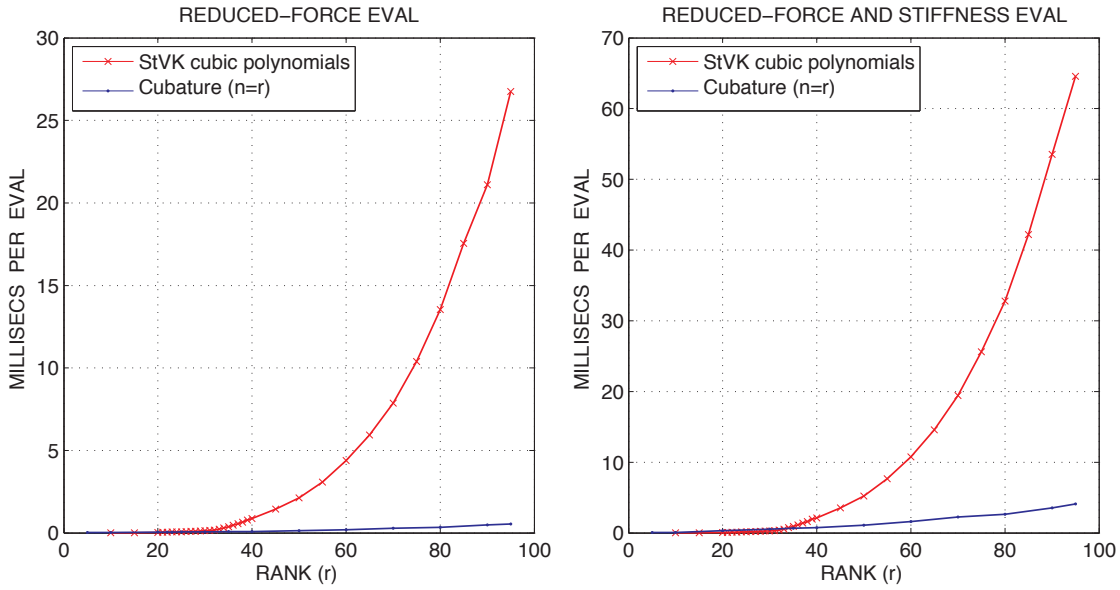


Figure 2.10: **Comparison to StVK reduced-force evaluation costs:** *We compared a St.Venant-Kirchhoff (StVK) polynomial model, versus a cubature-based approximation using $n=r$ elements. Our $O(r^2)$ reduced-force evaluation becomes faster around $r=23$, and scales substantially better that the $O(r^4)$ StVK polynomial model. When evaluating both reduced forces and stiffness matrices ($O(r^3)$), the cubature scheme becomes faster around $r=35$.*

interactive rates, allowing the user to control the amount of air pressure[2] interactively.

Haptic force-feedback rendering can exploit reduced-order models for simulation speed, and output-sensitive collision processing [9]. Optimized cubature enables complex nonlinear material models for real-time haptic rendering applications. As a proof of concept, we simulated a hollow rubber-like structure using both StVK and Mooney-Rivlin material models; without loss of generality, linear modal analysis (LMA) was used to generate an $r=10$ shape basis. The Mooney-Rivlin material gives clearly different behavior than the StVK model (see Figure 2.14).

[2]Pressure forces were modeled as a linear function of the form, $\mathbf{Pq}+\mathbf{f}_0$, where the constant matrix, $\mathbf{P}\in\mathbb{R}^{r\times r}$, and offset $\mathbf{f}_0$ model the subspace force resulting from a unit pressure applied to an internal tetrahedron's face whose deformed normal is approximated by transforming the material-frame normal by the deformation gradient (a quantity linear in $\mathbf{q}$).

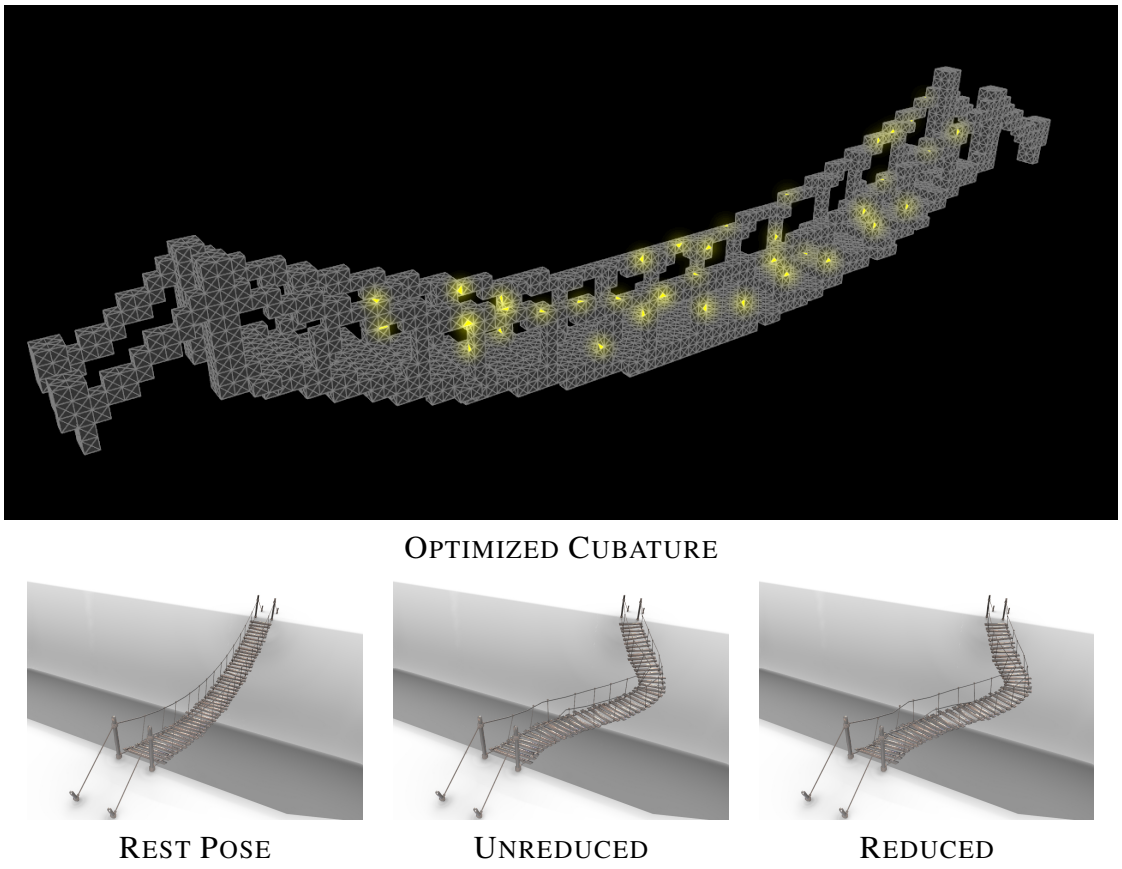


Figure 2.11: **Optimizing cubature for an embedded rope bridge:** *(Top) Rasterized polygon-soup bridge with cubature tetrahedra/points highlighted. (Bottom-Left) The undeformed bridge is subjected to a sideways impulse at its midpoint, and simulated using the (Bottom-Middle) unreduced implicit Newmark integrator, and the (Bottom-Right) implicit Newmark subspace integrator. Optimized cubature achieved 200 timesteps/second (implicit Newmark subspace) whereas unreduced implicit Newmark (with PARDISO solver) achieved 0.25 timesteps/sec.*

Hyperelastic material test: To investigate the ability of optimized cubatures to approximate nonlinear material response, we performed virtual compression tests on cubes (see Figure 2.15) made using three hyperelastic constitutive models: St. Venant-Kirchhoff, Mooney-Rivlin and Arruda-Boyce (see Appendix 2.6.3). In each case we precompute a compression test to estimate a PCA basis and reduced model for optimization and simulation[3]. Results of the compression tests are in Figure 2.13, and illustrate that optimized

[3]We compress a cube of the material by constraining the nodes of the top-face and lowering them by a small amount in a quasi-static simulation. When the material response reaches equilibrium, we

Figure 2.12: **Reanalysis of balloon inflation** *using cubature optimized from PCA-based simulation data. The dynamic Mooney-Rivlin rubber balloon can be inflated interactively.*

cubature can accurately reproduce nonlinear force responses when given a suitable subspace deformation model. For the StVK sample, our parameters were $\lambda = 1000$ and $\mu = 5000$. For the Arruda-Boyce, we used $\mu = 5000$, $N = 5$, and $K = 1 \times 10^5$. For the Mooney-Rivlin sample, we used $\mu_{10} = 1 \times 10^5$, $\mu_{01} = 10$, and $K = 1 \times 10^5$.

Nonlinear modal sound synthesis: Linear modal analysis is widely used for vibration modeling of effectively rigid objects since the runtime space and time complexity of integrating r modes with an IIR filter is only $O(r)$ flops [70], after which the modal coef-

record the total upward forces on the top-face nodes and record the displacement state. We then perform PCA on these recorded states to produce a linear basis, use these states to train a cubature rule, and observe the upward forces reproduced by the cubature rule. Because the top-face nodes are constrained in the original simulation, we need to artificially include them in the PCA basis for training cubature and reduced simulation. If $\mathbf{U}'$ is the basis produced by PCA of the simulation data, then the basis we use for the reduced simulation is $\mathbf{U} = [\mathbf{U}', \mathbf{u}_{top}]$, where $\mathbf{u}_{top}$ is a normalized $3N$ basis vector with positive vertical displacement for the top-face nodes. This allows us to measure the total upward force on the top-face nodes reproduced by the cubature force model by unprojecting the reduced force using $\mathbf{U}$.

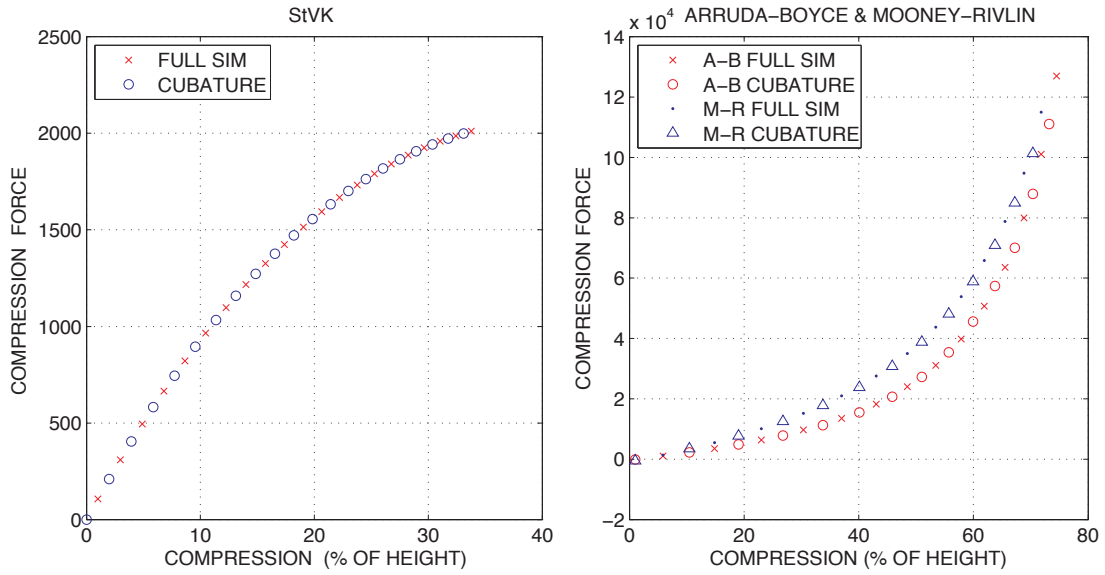


Figure 2.13: **Virtual compression test:** *The force response reproduced by optimized cubature closely matches the full simulation for all compression amounts tested. The Arruda-Boyce and Mooney-Rivlin materials could not be compressed beyond 85% due to element inversion, whereas the St.Venant-Kirchhoff (StVK) material softened significantly at about 35% compression, ultimately leading to inversion and stiffness matrix indefiniteness.*

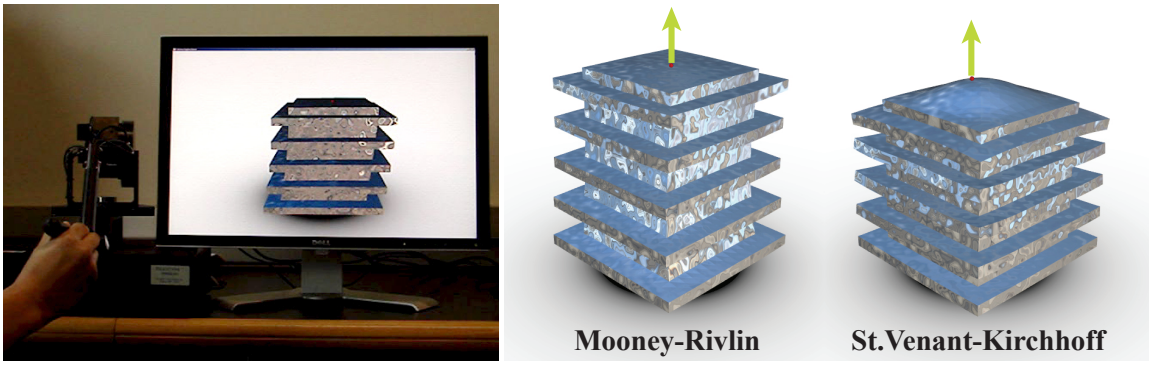


Figure 2.14: **Haptic force-feedback rendering examples:** *The implicit Newmark subspace integrators could deliver 5000 Hz rates.*

ficients **q** can be used to evaluate sound radiation [67]. However, for objects such as thin shells, even small deformations can induce nonlinear dynamics, and effectively "coupling" linear modes. To avoid the N-dependent costs of a fully nonlinear vibration analysis, such as in [102], we apply cubature optimization to nonlinear StVK forces based on sampling the space of linear-mode shapes. Optimized cubature provides an efficient

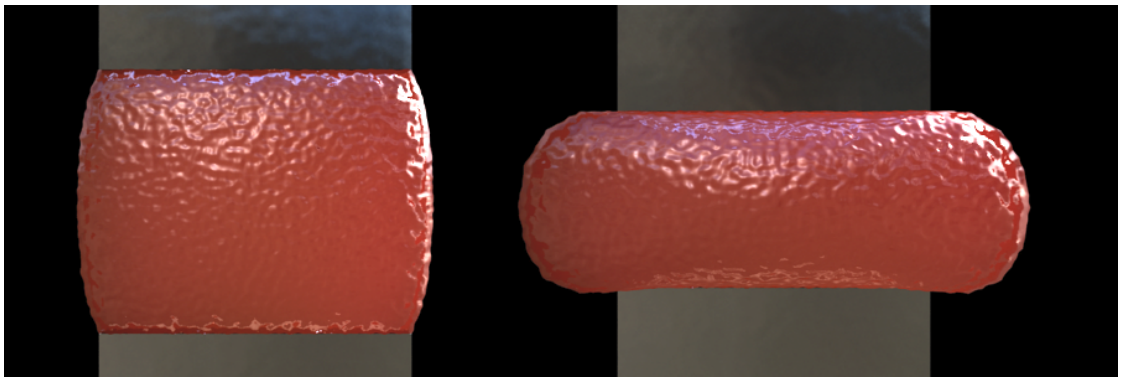


Figure 2.15: **Compressing the Arruda-Boyce material cube**

$O(r^2)$ evaluation of subspace forces, suitable for long-time explicit Newmark subspace integration [78]. Subspace integration provides easy access to modal coordinates $\mathbf{q}(t)$ used in sound radiation models (here we use the far-field, low-frequency, monopole radiation model (see (15) in [67])). Results are shown in Figure 2.2, and the accompanying video (http://www.cs.cornell.edu/Projects/Sound/cubature/). We compared our cubature approximation to a brute-force subspace simulation [78] and obtained nearly identical sounds. However, computing $\mathbf{f}(\mathbf{q})$ by evaluating $O(N)$ unreduced forces, followed by subspace projection (multiplication by $\mathbf{U}^T$), at each explicit timestep was approximately $110\times$ more expensive; 16-core parallelization was used to compute the comparison. In Chapter 3, we expand on this application by applying it to reduced thin shell models.

2.8 Conclusion and Discussion

In summary, optimized cubature is a simple and mathematically sound way to build reduced-order force models for subspace integration. It is not restricted to any particular shape model, and it supports various hyperelastic material models. Cubature can be trained for geometrically complex examples, and fast subspace integration performance

can enable interactive graphics, large multibody simulations, or simulations requiring high temporal rates such as haptics and sound synthesis. Although cubature schemes depend on training data, they also are surprisingly robust to over-fitting.

Discussion and Limitations: We have provided some examples to provide evidence in support of cubature optimization, however there are many other ways to realize its benefits, and also other limitations to overcome.

Exploiting sparsity is important for fast evaluation and training of high-rank r models, whereas our current analysis is limited to models with dense $\mathbf{U}$ bases. Generating sparse bases and cubature schemes can lead to linear-time $O(r)$ schemes for reduced force and sparse stiffness matrix evaluation. The existence of an $O(r)$ algorithm for reduced force evaluation using dense rank-r displacement bases remains an open problem.

Although cubature schemes support general subspace deformations, we have mostly considered linear (constant $\mathbf{U}$) shape models here, however many other successful models exist, e.g., modal derivatives [8]. We have also constructed cubature schemes for articulated and skinned mesh models [72], however such models require more efficient implicit Newmark subspace integrators with efficient sparse matrix solves and/or preconditioning for similar performance. Physically based character animation, especially detailed skin deformation and facial animation, are areas likely to benefit from fast cubature schemes. Some shape models may be more susceptible than others to element inversion, and non-element-based approaches for evaluating $\mathbf{g}$ may help. For example, we implemented subspace deformations based on modal warping [32], however we found element-based cubature schemes susceptible to element inversion, presumably due to highly extrapolated element deformations.

Future work includes applying subspace-based cubature optimization to the simulation of physical phenomena other than volumetric deformable objects, e.g., shells, MEMS, etc. Efficient error estimators allows the possibility of adaptive simulation. For general kinematics, the mass matrix can be time dependent and potentially expensive to evaluate. Optimized cubature might also be used for fast estimates of the mass matrix, $\int_\Omega U(\mathbf{x})^T U(\mathbf{x}) \rho(\mathbf{x}) d\Omega$.

Finally, we have proposed a greedy algorithm for cubature optimization, however, a stronger theoretical footing is desirable for automatic cubature generation. For example, it is tempting to think of cubature as some sort of critical points of some class of functions, similar to the definition of Gaussian quadratures as the zeros of Legendre polynomials and other functions. It appears that similar accuracy may be possible using half as many cubature points if Gaussian-quality cubatures could somehow be learned, thus leading to a two-fold speedup in reduced force evaluation.

CHAPTER 3

A PRACTICAL NONLINEAR SOUND MODEL FOR NEAR-RIGID THIN SHELLS USING CUBATURE

We propose a procedural, physics-based method for synthesizing realistic sounds due to nonlinear thin-shell vibrations. We use linear modal analysis to generate a small-deformation displacement basis, then couple the modes together using nonlinear thin-shell forces. To enable audio-rate time-stepping of mode amplitudes with mesh-independent cost, we propose a reduced-order dynamics model based on a thin-shell cubature scheme, building on the method presented in Chapter 2. Familiar examples are presented including rumbling trash cans and plastic bottles, crashing cymbals, and noisy sheet metal objects, each with increased richness over linear modal sound models. This work was originally published as part of Chadwick et al. [23].

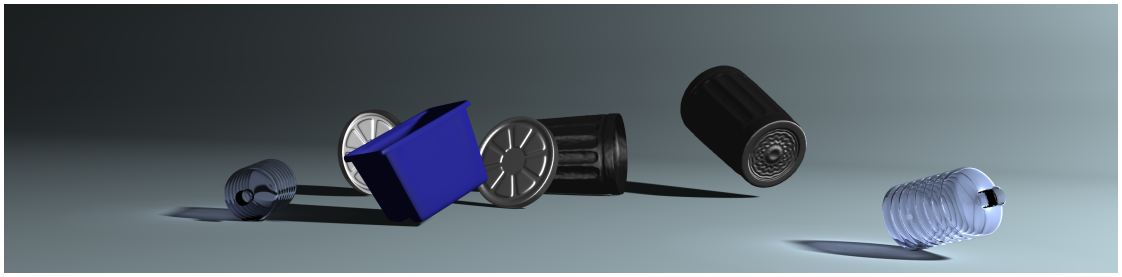


Figure 3.1: **Crash!** *Our physically based sound renderings of thin shells produce characteristic "crashing" and "rumbling" sounds when animated using rigid body dynamics. We synthesize nonlinear modal vibrations using an efficient reduced-order dynamics model that captures important nonlinear mode coupling.*

3.1 Introduction

Linear modal sound models are widely used for rigid bodies in computer animation and virtual environments [140, 103, 16], and when combined with acoustic transfer models for sound radiation [67] they can provide convincing physically based sound sources, especially for pure ringing tones such as chimes, bells, or "knocks." Unfortunately, we lack effective sound models for a broad class of noisy virtual objects: thin shells (objects with thicknesses orders of magnitude smaller than their other dimensions). Thin shells are very common in real and virtual environments, and produce rich and easily recognizable impact sounds: sheet metal objects (trash cans, oil drums, tin roofs, machinery), plastic containers (water bottles), musical instruments (cymbals), etc. Their rich nonlinear vibrations produce proverbial "crashes" and "rumbles" that are poorly approximated by linear modal sound models which lack nonlinear mode coupling. To make matters worse, thin shells are often very loud and important sound sources due to their ability to vibrate and radiate sound so effectively, e.g., consider a metal roof pelted by hail. Alas, their expensive nonlinear dynamics have made thin shells computationally impractical for physically based sound synthesis.

In this paper, we propose an efficient method for synthesizing realistic sounds from thin-shell structures undergoing small but nonlinear vibrations. Given a description of an object's geometry and material properties, we compute linear vibration modes, then couple these modes together using the nonlinear thin-shell force model. To accelerate nonlinear modal dynamics, we optimize a thin-shell cubature scheme to evaluate reduced-order shell forces at costs independent of the geometric complexity of the model. We show that the complex internal dynamics of thin-shell models can be approximated with sufficient accuracy and efficiency to allow practical synthesis of plausible

thin-shell sounds. We also address sound-related locking effects that arise when simulating nonlinear modal dynamics that might produce pitch-glide artifacts in general animations.

Other Related Work: Thin elastically deformable models have seen widespread use in computer animation, especially for large-deformation simulations of parameterized cloth models [133, 6]. Plate-like cloth models with flat rest configurations are most common, however recently shell models for objects with instrinsically curved rest configurations have gained attention, e.g., to model clothing with folds and wrinkles [17]. While various thin-shell models exist in the mechanics literature (see [27]), discrete shell models that better meet computer animation needs have appeared recently that use simple bending energy formulations that allow standard cloth solvers to produce convincing large-deformation thin-shell dynamics [52, 17]; we use the elastic thin-shell model summarized in Gingold et al. [48]. Other works have aimed to make shell simulations faster and better [34, 33, 51]. The mathematical structure of bending energy formulations has been exploited and simplified to accelerate near-isometric deformation of thin plates [12] and shells [46]. Our work differs in that we do not require large-deformation animations or sophisticated linear system solvers, but rather focus on small mode-related shell vibrations and try to obtain explicit time-stepping costs sublinear in geometric complexity to enable audio-rate sound synthesis. We also prefer mode-based representations since they are preferred for auralization with frequency-domain acoustic transfer models.

Linear modal vibration models are well known in animation [106, 70]. They have proven effective for procedural sound synthesis, and are increasingly used to simulate rigid-body impact sounds [2, 35, 138, 140, 103] in part due to excellent synthesis speed for

interactive applications [140, 112, 16]. “Modal warping” has been used to approximate large-deformation thin shells for animation [31], but such models are of limited use for nonlinear sound synthesis since the underlying linear modal oscillators are uncoupled by design.

There is a tremendous amount of work on the nonlinear vibrations of plates and shells in engineering [99, 98] and related structure borne sound [38]. Important examples are nonlinear vibrations in musical instruments [44], such as (the notoriously difficult to simulate) gongs and cymbals [26]. Simple all-pass nonlinear passive filters have been used to mimic nonlinear mode-coupling effects [108]. For nonlinear modal analysis, linear eigenmodes are often used for nonlinear subspace integration of dynamics to resolve weak material nonlinearities in small-strain configurations and resolve mode-mode coupling [10] (for a good discussion on nonlinear mode coupling in beams and plates see [87]). Nonlinear normal modes [100, 135] have been used to describe a nonlinear vibration mode’s shape as a linear superposition of other linear modes, i.e., to resolve nonlinear mode coupling. However, likely due to the high computational complexity of simulating nonlinearly coupled modal models (often $O(r^4)$ or worse), we are unaware of sound synthesis results in the literature that demonstrate results comparable to ours, i.e., with several hundred fully coupled modes. We achieve this by extending cubature optimization techniques of An et al. [4]; their method estimates volumetric cubature schemes, and was even used to evaluate nonlinear modal shell vibrations for sound synthesis, but the thin shell had to be modeled using several hundred thousand tetrahedral elements. We extend cubature schemes to handle thin shells more efficiently, and obtain cubature-based reduced-order modal models with $O(r^2)$ time-step complexity for r nonlinearly coupled modes.

The only physically based sound rendering work in graphics that addresses nonlinear object vibrations is O'Brien et al. [102]. They use an explicitly integrated large-deformation finite element model to simulate nonlinear vibrations of objects using small time-step sizes, and a time-domain ray-based "Rayleigh method" (a.k.a. direct propagation) to approximate sound radiation. Interesting results were obtained for short animations with large deformations and buckling. Unfortunately simulation times were on the order of a day (circa 2001) for models of rather modest geometric complexity ($<$2000 tetrahedra), and the expensive radiation model is known to have limited accuracy [40]. Bilbao has considered energy conserving finite difference discretizations and time-stepping schemes for nonlinear plates to generate plausible sounds [13]. In contrast to these works, we focus on efficient sound models for nonlinearly forced thin-shell vibrations: we develop efficient subspace integration techniques for geometrically complex nonlinear modal models which drive frequency-domain acoustic transfer models that are consistent with frequency-domain wave radiation.

3.2 Background: Thin-Shell Dynamics

We consider discrete thin-shell models on manifold with boundary triangle meshes with N_Δ triangles, and N_v vertices. Following a suitable discretization via the finite element (or other) method we obtain an N-dimensional system of ordinary differential equations,

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{f}_{int}(\mathbf{u}) = \mathbf{f}_{ext} \tag{3.1}$$

where $\mathbf{u} = \mathbf{u}(t) \in \mathbb{R}^N$ are mesh vertex displacements, $\dot{\mathbf{u}}$ are velocities, $\ddot{\mathbf{u}}$ are accelerations, $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the mass matrix; $\mathbf{f}_{int}(\mathbf{u})$ describes nonlinear internal thin-shell forces; and $\mathbf{f}_{ext}$ are time-dependent external forces such as gravity or contact forces.

For lightly damped small vibrations, we use linear Rayleigh damping [10] with $\mathbf{D} = \alpha\mathbf{M} + \beta\mathbf{K}$, with $\mathbf{K}$ the stiffness matrix (Jacobian of $\mathbf{f}_{int}$) evaluated at $\mathbf{u} = 0$.

Without loss of generality, we use the elastic shell model of [48], in part because it is based on physical material parameters that simplify parameter tuning for sound synthesis. The deformation strain energy, $E(\mathbf{u})$, is an integral over the surface of the strain energy density, $W(\mathbf{u};X)$, where X is a material position on the undeformed surface. The strain energy density is decomposed in two parts

$$W = W_m + W_b \tag{3.2}$$

where W_m is the membrane strain energy density which penalizes tangential stretching or compression; and W_b is the bending energy density which resists bending away from the rest configuration. The membrane and bending strain energy densities are defined in terms of per-triangle strain tensors:

$$W_m = \frac{Y\,h}{2(1-\nu^2)}\,\left[(1-\nu)\,\mathrm{tr}(\boldsymbol{\varepsilon}_m^2) + \nu\,\mathrm{tr}(\boldsymbol{\varepsilon}_m)^2)\right] \tag{3.3}$$

$$W_b = \frac{Y\,h^3}{24(1-\nu^2)}\left[(1-\nu)\,\mathrm{tr}(\boldsymbol{\varepsilon}_b^2) + \nu\,\mathrm{tr}(\boldsymbol{\varepsilon}_b)^2)\right] \tag{3.4}$$

where h is the shell thickness, Y is Young's modulus, and ν is Poisson's ratio (see Gingold et al. [48] equations for $W_{membrane} = W_m$ and $W_{bending}^{Koiter} = W_b$). The membrane and bending strains are $\boldsymbol{\varepsilon}_m$ and $\boldsymbol{\varepsilon}_b$, respectively; $\boldsymbol{\varepsilon}_m$ is a 3×3 tensor which varies as triangle edges deviate from their rest lengths; and $\boldsymbol{\varepsilon}_b$ is a 3×3 tensor which changes as the dihedral angles between a triangle and its three neighbors vary from the corresponding dihedral angles in the shell's rest pose (see [48] for definitions). Consequently, the strains and strain energy density, W, can be defined as piecewise constant over shell triangles, with each triangle's W_m value a function of the triangle's 3 vertex positions, whereas its bending strain energy, W_b, is a function of 6 vertices (see Figure 3.2).

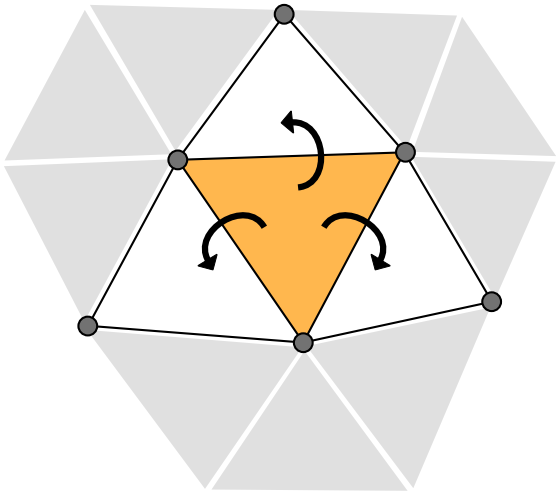


Figure 3.2: **Stencil of triangle element**

It follows that the deformation energy is given by the integral over the undeformed surface, S:

$$E(\mathbf{u}) = \int_S W(\mathbf{u};X)\, dS_X = \sum_{i=1}^{N_\Delta} A_i W_i(\mathbf{u}) \tag{3.5}$$

where W_i is the piecewise constant strain energy density for triangle i, and A_i is its area. The desired internal thin-shell force is

$$\mathbf{f}_{int} = \nabla_{\mathbf{u}} E(\mathbf{u}) = \sum_{i=1}^{N_\Delta} A_i \nabla_{\mathbf{u}} W_i(\mathbf{u}). \tag{3.6}$$

which gathers both membrane and bending contributions.

3.3 Nonlinear Modal Sound Synthesis for Thin Shells

3.3.1 Reduced-order Thin-Shell Dynamics

We now describe the nonlinear mode-coupled dynamics model used for sound synthesis.

Linear Modal Analysis Basics: As a first step, we compute r linear eigenmodes of the thin-shell system (3.1) using standard methods [126, 10]. We compute undamped vibration modes by solving the generalized eigenvalue problem,

$$\mathbf{K}\mathbf{u}_j = \omega_j^2 \mathbf{M}\mathbf{u}_j, \qquad j = 1 \dots r, \tag{3.7}$$

where the j^{th} displacement eigenmode, $\mathbf{u}_j$, corresponds to a vibration at the j^{th} (smallest but nonzero) natural frequency, ω_j. We compute the first r modes corresponding to nonzero eigenvalues ("rigid body modes" with zero eigenvalues are discarded). The eigenvectors are assembled in the mode matrix, $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_r] \in \mathbb{R}^{N\times r}$ where N is the number of degrees of freedom in the system ($r \ll N$). Without loss of generality we "mass normalize" the eigenvectors so that

$$\widetilde{\mathbf{M}} = \mathbf{U}^T\mathbf{M}\mathbf{U} = \mathbf{I} \quad \text{and} \quad \widetilde{\mathbf{K}} = \mathbf{U}^T\mathbf{K}\mathbf{U} = \Lambda = \text{diag}(\boldsymbol{\omega}^2). \tag{3.8}$$

The linear modal model assumes vertex displacements are given by a linear superposition of mode shapes, $\mathbf{u}(t) = \mathbf{U}\mathbf{q}(t)$, where $\mathbf{q}(t) \in \mathbb{R}^r$ are the generalized modal coordinates.

Nonlinear Subspace Integration: To obtain nonlinear coupling between linear modes we employ dimensional model reduction [10, 78] by substituting $\mathbf{u} = \mathbf{U}\mathbf{q}$ into the full-dimensional equations of motion (3.1), and premultiplying by $\mathbf{U}^T$ to project into the r-dimensional modal subspace:

$$\ddot{\mathbf{q}} + \widetilde{\mathbf{D}}\dot{\mathbf{q}} + \widetilde{\mathbf{f}}_{int}(\mathbf{q}) = \widetilde{\mathbf{f}}_{ext} \tag{3.9}$$

where

$$\widetilde{\mathbf{D}} = \alpha\mathbf{I} + \beta\Lambda \tag{3.10}$$

$$\widetilde{\mathbf{f}}_{int}(\mathbf{q}) = \mathbf{U}^T\mathbf{f}_{int}(\mathbf{U}\mathbf{q}) \tag{3.11}$$

$$\widetilde{\mathbf{f}}_{ext} = \mathbf{U}^T\mathbf{f}_{ext}. \tag{3.12}$$

This equation provides nonlinear mode coupling when integrated (see Figure 3.3). Unfortunately explicit time-stepping of these nonlinear equations at audio rates (44.1 kHz; $\Delta t \approx 2.3 \times 10^{-5}$) is quite expensive due to $\widetilde{\mathbf{f}}_{int}$ evaluation, which involves subspace-projection of $\mathbf{f}_{int}$ in (3.6) via a gather over N_Δ triangles—an undesirable $O(rN_\Delta)$ cost per audio timestep.

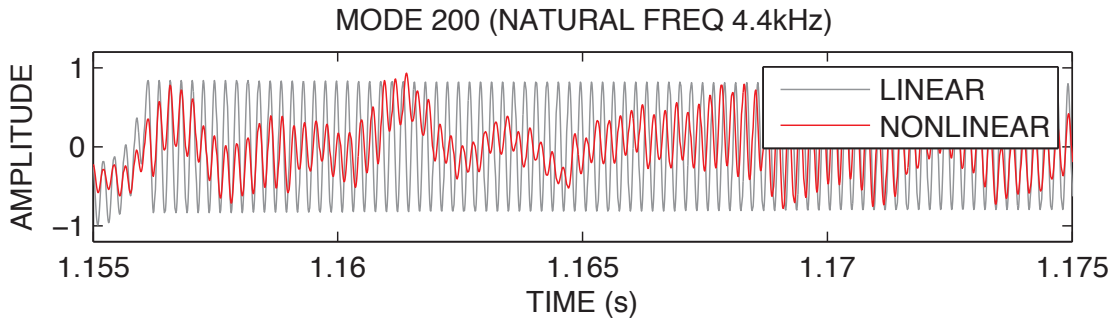


Figure 3.3: **Nonlinear mode coupling:** *The nonlinear and linear modal dynamics of $q_{200}(t)$ are compared for the ride cymbal's response to a single metal ball impact (the first video example). The nonlinear mode exhibits rich dynamics, and strong coupling to lower frequency modes.*

In the linearized small-deformation case this bottleneck disappears since the modal equations decouple: $\mathbf{f}_{int} = \mathbf{K}\mathbf{u}$ so that $\widetilde{\mathbf{f}}_{int} = \Lambda\mathbf{q}$. Each mode can be integrated efficiently using an IIR filter [55] (e.g., using filter coefficients in [70]) so that each explicit time-step has only $O(r)$ cost, but nonlinear mode coupling is lost.

3.3.2 Thin-Shell Cubature Scheme

We accelerate $\widetilde{\mathbf{f}}_{int}$ evaluation by extending the volumetric approach of An et al. [4] to optimize cubature schemes for thin shells. We use a compound cubature scheme that evaluates both bending and stretching force integrals simultaneously thereby sharing

vertex-deformation computations. We make the approximation

$$\widetilde{\mathbf{f}}_{int}(\mathbf{q}) = \mathbf{U}^T \mathbf{f}_{int}(\mathbf{U}\mathbf{q}) = \sum_{i=1}^{N_\Delta} A_i \, \mathbf{g}_i(\mathbf{q}) \tag{3.13}$$

$$\approx \sum_{i \in \mathcal{C}} w_i \, \mathbf{g}_i(\mathbf{q}), \tag{3.14}$$

where $\mathbf{g}_i(\mathbf{q}) \equiv \mathbf{U}^T \nabla_{\mathbf{u}} W_i(\mathbf{U}\mathbf{q})$, and w_i are cubature weights, and $\mathcal{C}$ are a set of integers defining cubature triangles (see Figure 3.4).

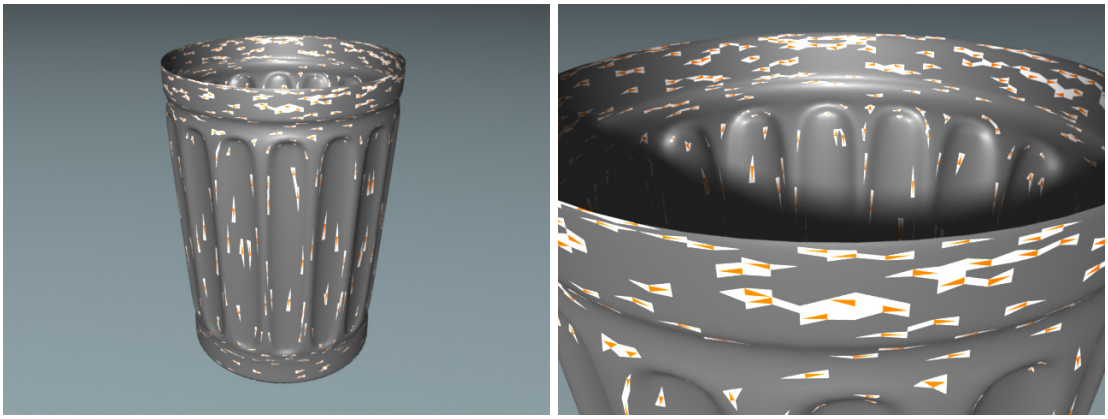


Figure 3.4: **Illustration of cubature scheme:** *(Left) Trash can (200 modes) with 800-feature cubature scheme; (Right) close-up of triangle-flap features.*

Cubature Training: Since the strain energy density (and thus force density) defined in the shell model is piecewise-constant over a triangle element, it suffices to only consider one cubature point per triangle during the optimization pre-process. However, at run-time the triangle and its three edge flaps must be reconstructed for each cubature point, as the bending force is dependent on the state of neighboring triangles. We also considered alternate cubature schemes, including optimizing separate weights for the two force integrals, but this did not significantly affect convergence behavior. To generate training samples for cubature optimization, we randomly sample a Gaussian for each mode to get physically plausible training poses [4]. Convergence plots are shown in Figure 3.5.

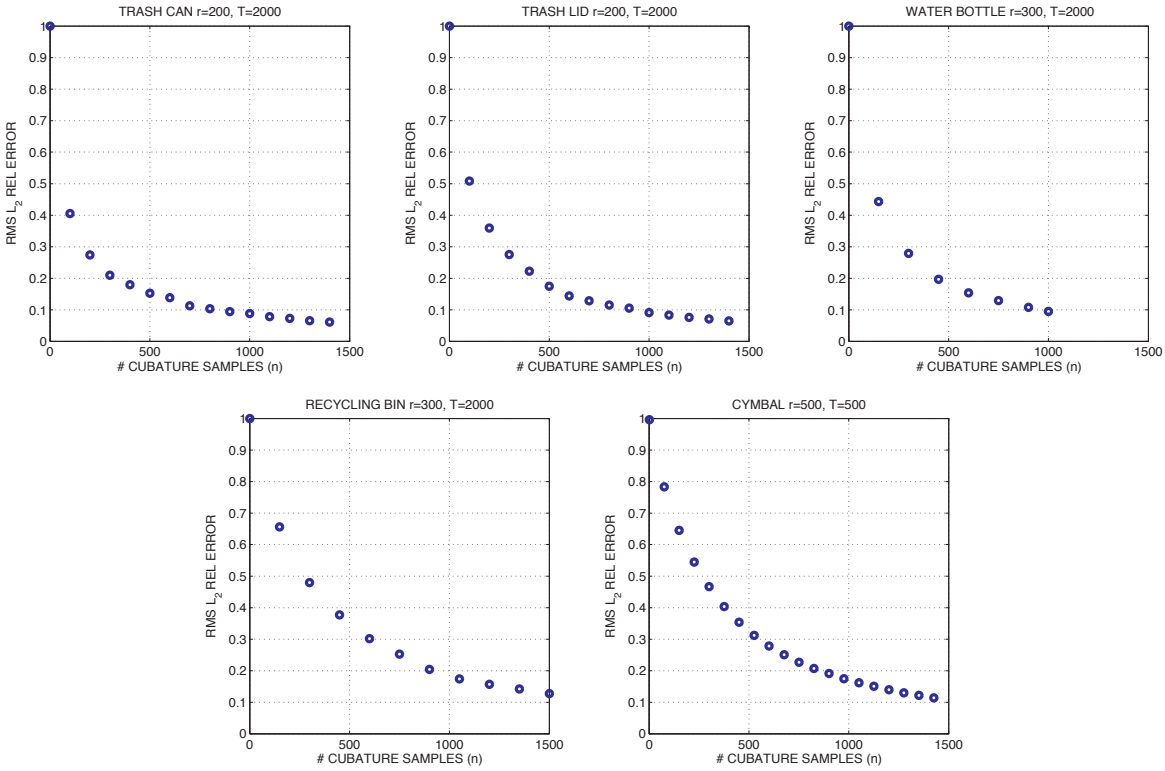


Figure 3.5: **Cubature training convergence plots** *reveal that* $\sim 10\%$ *error is often obtained after* $n = 4r$ *cubature features, which is higher than the volumetric modal models in [An et al. 2008].*

$O(r^2)$ **Force Evaluation:** In practice, we can construct cubature schemes to modest accuracies (e.g., 10% relative error) which have the number of cubature samples, $n = |\mathcal{C}| = O(r) \ll N_\Delta$, so that the thin-shell cubature scheme can approximate $\widetilde{\mathbf{f}}_{int}(\mathbf{q})$ at $O(r^2)$ cost. More important than complexity is that the scheme is fast in practice. Achieving full nonlinear coupling of r modes at $O(r^2)$ cost makes offline sound synthesis practical for several hundred modes (see Table 3.2). To understand this cost, note that each $\mathbf{g}_i(\mathbf{q})$ r-vector in (3.14) can be evaluated in $O(r)$ flops using three steps: given the triangle element's 6 stencil vertices, V (see Figure 3.2), we (1) evaluate the 6 vertex displacements, $\mathbf{u}_V = \mathbf{U}_V \mathbf{q}$ in $O(r)$ flops; (2) evaluate the 6 vertex forces $\mathbf{f}_V = \nabla_{\mathbf{u}_V} W_i(\mathbf{u}_V)$ in $O(1)$ flops, then (3) project the vertex forces into the r-dimensional subspace, $\mathbf{g}_i = \mathbf{U}_V^T \mathbf{f}_V$, in $O(r)$ flops.

3.3.3 Modal Sound Synthesis

Given a method for integrating the subspace dynamics, as presented in the previous section, we can synthesize sound synchronized to a rigid body physics simulation. The simulation provides us with external forces $\mathbf{f}_{ext}$, which can be projected into reduced forces $\tilde{\mathbf{f}}_{ext}$, and so we can perform time-integration of the system to obtain a time series of modal coordinates $\mathbf{q}(t)$. We now show how to produce sound from this time series.

If we assume that each mode affects the surrounding medium independently (even though their dynamics may be coupled), then the final pressure signal $s(t)$ for a relative listening position x can be obtained as:

$$s(t) \quad = \quad \sum_{k=1}^{r} |p_k(x)| q_k(t) \tag{3.15}$$

Here, $p_i(x)$ is the *acoustic transfer function*, which gives the magnitude of mode i's effect on the pressure at location x. This can account for cases where certain modes do not radiate effectively towards certain locations due to the geometry of the object. A bell, for example, sounds very different depending on the listening location. O'Brien et al. [103] proposes a simple far-field monopole approximation, essentially treating the object as a pulsating sphere. Other models account for more variation and accuracy [67, 144]. Please see the full paper of Chadwick et al. [23] for details on how transfer was handled for the case of thin shells.

Model	L (m)	tri	vtx	N	modes	freq (kHz)	material
Trash Can	0.75	77536	38833	116499	200	0.071 – 4.43	Steel
Trash Lid	0.55	34312	17286	51858	200	0.112 – 6.79	Steel
Water Bottle	0.46	28658	14418	43254	300	0.116 – 3.59	Polycarb.
Recycling Bin	0.61	109568	54945	164835	300	0.062 – 2.21	Polycarb.
Cymbal	0.50	61952	31104	93312	500	0.061 – 9.94	Bronze

Model	ν	Y (GPa)	h (mm)	α	β (10^{-9})	n_{cuba}	Error$_{cuba}$	kL
Trash Can	0.30	190	2	0.5	75	800	10.3%	0.98 – 61
Trash Lid	0.30	190	2	0.5	75	800	11.5%	1.1 – 68
Water Bottle	0.37	2.4	2.25	0.5	400	900	10.7%	0.98 – 48
Recycling Bin	0.37	2.4	5	4.0	300	1200	15.7%	0.70 – 30
Cymbal	0.33	124	0.7	1.0	6.25	1500	10.7%	0.57 – 92

Table 3.1: **Model Statistics**. *The timestep Δt was 1/44100 seconds for all examples except the cymbal, where it was 1/88200.*

3.4 Results

We now describe numerical and sound experiments for several models and multibody collision scenarios. For each example, we provide comparisons between the linear and nonlinear vibration models. In each case, the sounds resulting from the linear model lack the "crash" and "rumble" effects exhibited by the nonlinear results. Please see our accompanying video (http://www.cs.cornell.edu/projects/harmonicshells/) for all animation and sound rendering results. Model statistics are provided in Table 3.1. Representative timings are given in Table 3.2.

Implementation Details: We precompute dominant linear vibration modes using Matlab's generalized eigenvalue solver (using ARPACK's shift-and-invert spectral transformation). To improve our graphics model's mesh quality for vibration and radiation analysis, we remesh the shells (using GNU GTS). All animations are performed using rigid body dynamics with vibration models defined in the appropriate rigid body

frame [125]. Collisions are detected using a rigid sphere-tree bounding volume hierarchy, and resolved using a linear Kelvin-Voigt penalty contact model. Rigid body dynamics are time-stepped using symplectic Euler at rates sufficient to resolve penalty contact forces; modal vibrations are time-stepped (explicit subspace Newmark) at audio rates (e.g., 44100Hz); A simple contact damping model is used to damp vibrations of objects in ground contact. In our simulation pipeline, we first simulate rigid-body motion and dump subspace force impulses to disk, then in a second pass we compute modal vibrations and synthesize sound at the listening position. Each object's final sound is computed as a linear superposition of modal contributions with a simple HRTF model, $H(\omega,\boldsymbol{x})$ [19]; $sound(\boldsymbol{x},t) = \sum_{k=1}^{r} |H(\omega_k,\boldsymbol{x})||p_k(\boldsymbol{x})|q_k(t)$. Graphics frames were rendered using Pixar's RenderMan. All floating point computations were performed using double precision.

Model	Modes r	Modal Analysis	Cubature Precomp.	Timestep Cost	Simulation Cost (per second of audio)
Trash Can	200	569 s	2.49 hr	16.1 ms	714 s
Trash Lid	200	170 s	1.87 hr	14.6 ms	642 s
Water Bottle	300	314 s	4.31 hr	23.6 ms	1026 s
Recycling Bin	300	2332 s	9.65 hr	27.8 ms	1224 s
Cymbal	500	1155 s	3.88 hr	44.3 ms	3900 s

Table 3.2: **Representative Timings:** *All timings are for a single 2.66GHz Xeon X5355 processor core, except "Cubature Precomp" which used 8 cores.*

EXAMPLE (Cymbal): We modeled a large ride cymbal (50cm diameter, bronze), which is known to be a challenging example for modal vibrations [26]. The linear modal model of a ride cymbal produces a very clean tone that sounds more like a smaller crash cymbal, and it is unable to produce the proverbial "crash" sound as well as the nonlinear model. Both models sound very plausible with acoustic transfer.

EXAMPLE (Trash Can with Lid): The nonlinear modal model produces a dramatic improvement in the sound of the trash can (and lid) relative to the linear modal model; the nonlinear model produces a characteristic "crashing" sound, whereas the linear model makes a "ding" sound. The spolling (spinning & rolling) of the trash can lid has a more distinctive sound than the linear model.

EXAMPLE (Water Bottle): We modeled a round 5-gallon water bottle out of polycarbonate plastic, and tuned damping parameters by comparing to informal experiments. The nonlinear sound model captures a characteristic drum-like fluttering after impact better than the linear sound model. A comparison to a real water bottle impact experiment is provided in the accompanying video, and produces a qualitatively similar sound.

EXAMPLE (Plastic Recycling Bin): While less dramatic than other examples, the nonlinear model captures a familiar "wobbling" sound which is missing from the linear model.

MULTIBODY EXAMPLES: We simulated several multibody collision scenarios to demonstrate the feasibility of our approach for computer animation (see Figure 3.6, and video results).

Stability: Unlike linear modal models which can be stably integrated with IIR filters, our nonlinear subspace vibration model can suffer time-stepping instabilities. Fortunately, subspace integrators are typically more stable than their unreduced counter-

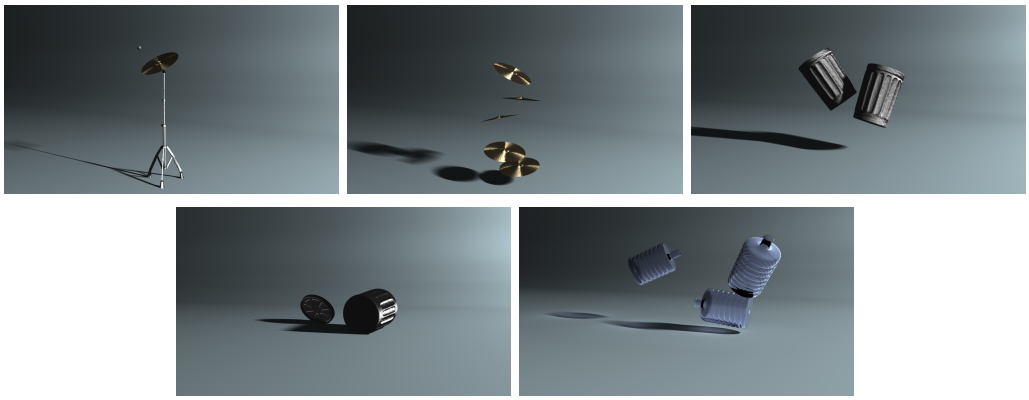


Figure 3.6: **Multibody collision scenarios** *were simulated for (from left to right) a cymbal with metal balls, multiple cymbals, two trash cans, a trash can and lid, and polycarbonate water bottles—as well as the teaser image (Figure 3.1).*

parts [78]. We observed that our explicit subspace Newmark integrator was stable at audio rates (44.1 kHz) for all examples, except the cymbal which we integrated at 88.2 kHz. In contrast, traditional explicit Newmark integration required an exceedingly small timestep to be stable, e.g., the water bottle required 11.025 MHz rates (or 250$\times$ the 44.1 kHz rate).

COMPARISON (reduced vs. unreduced simulation): For validation, we compared water bottle impact sounds from our reduced-order model ($\Delta t = 1/44100s$) to those of nonlinear vibrations simulated in a full, unreduced setting via an explicit Newmark integrator ($\Delta t = 1/11025000s$ for stability)—implicit Newmark (with full Newton solves) was less competitive in our experiments. Although the unreduced model produced richer tones at higher amplitude impacts, both sounds were comparable and more interesting than pure linear vibrations. Unfortunately, while the reduced-order model took roughly 17.1 minutes to compute 1 second of sound (1026$\times$ slower than real time), the unreduced approach took 89.8 hours per second of sound (323,000$\times$ slower than real time).

Details of unreduced computation: Given the unreduced displacement $\mathbf{u}$ of the object, modal amplitudes are obtained via projection with the basis $\mathbf{q} = \mathbf{U}^T\mathbf{u}$ so that any of the previously discussed radiation methods may be applied. Given that the simulated model is unconstrained, we take steps to avoid rigid body motions in the unreduced simulation as these will result in errors in the modal projection. A $N \times 6$ "rigid basis" matrix $\mathbf{U}_R$ is constructed out of the rigid modes (those corresponding to eigenvalue 0) computed in Equation 3.7. This is used to produce a 6×6 "rigid mass" matrix $\mathbf{M}_R = \mathbf{U}_R^T\mathbf{M}\mathbf{U}_R$. Given the external forces acting at each time step, the component of acceleration resulting in rigid motion is identified as $\mathbf{U}_R\mathbf{M}_R^{-1}\mathbf{U}_R^T\mathbf{f}_{ext}$ and subtracted from the total acceleration vector. This allows the mesh to vibrate freely in place without undergoing rigid translation and rotation.

COMPARISON (different cubature errors): We simulated nonlinear modal models with different cubature errors to informally demonstrate their respective sound behavior. See the video for a comparison of the trash can simulated with cubature errors (and timestep costs) of 15.3% (10.5ms), 10.3% (16.1ms), 6.1% (27ms), and using brute-force subspace integration [78] we simulated 0% (166.7ms). We find that even cubature schemes with large relative error provide a significant qualitative improvement in sound quality over the linear model. Our cubature schemes are chosen to provide a tradeoff between sound quality and evaluation speed.

3.5 Conclusion

We have presented a practical method for generating plausible impact sounds for thin shells. By leveraging reduced-order modeling, we can produce nonlinear modal models that enable simulation of hundreds of vibration modes with fully coupled nonlinear modal dynamics. Compared to linear modal sound models, our objects produce more characteristic "crashing" and "rumbling" sounds. Please see the original publication of Chadwick et al. [23] for other important aspects of the method, such as the impulse limiter and the FFAT maps.

Limitations and Future Work: There are numerous ways to improve this result in future work. Simulating the full range of audible "all-frequency" sound poses numerous challenges, not only for precomputing thousands of vibration modes but also for coupling them together. The $O(r^2)$ complexity of the nonlinear modal model reflects the intrinsic complexity of simulating r coupled modes, but a near-linear-time subspace force algorithm would be a big breakthough for all-frequency nonlinear sound synthesis, especially since the modal amplitudes are needed for transfer-based rendering. In all cases, we would have liked to have used more vibration modes to produce "all-frequency" sound renderings; large models can also require many modes. The shell-based cubature schemes exhibit slower convergence rates than volumetric models [4], and higher accuracy and more scalable methods are required for faster and/or more complex models. We use a simplified rigid-body contact model, but collision processing should account for object vibrations (ideally in a reduced-order manner [71]) to properly capture chattering.

Beyond thin shells, how to devise efficient methods for evaluating all-frequency non-

linear vibration-based sound is an open problem. Modal locking reflects the limitations of the linear modal shape model, and we need more expressive shape models to handle difficult nonlinear and noise-like pheonomena; a fully developed (chaotic) cymbal crash is currently beyond the capability of our reduced-order vibration model, although the cymbal's acoustic transfer model appears plausible. Preliminary experiments with thin-shell models using nonlinear shape functions based on modal warping did not provide a significant improvement [31]. More generally, we need methods to evaluate accurate sound for large-deformation animations. Buckling is also a challenging nonlinear phenomenon which can produce significant sound radiation, e.g., crumpling paper.

CHAPTER 4

COMPRESSING MODAL DISPLACEMENT FIELDS FOR SOUND SYNTHESIS

Sound models based on pre-computed vibration modes, as used for rigid bodies [103] and near-rigid thin shells (Chapter 3), can have high memory costs when used at run-time. In order to determine which modes are excited by a given external force, the displacement fields of the modes must be queried. These displacement fields can take dozens or even hundreds of megabytes to store in memory per unique object. This chapter presents methods for compressing them based on mesh simplification and the exploitation of symmetry.

4.1 Introduction

Rigid body physics for animation and virtual environments has been an important and well-studied area of computer graphics, and relatively recently, much attention has been given to producing physically-based sound to enhance the visuals. They have been used to complement haptic feedback [104], synthesize sounds for fracture animations [144], and even approximate non-linear thin-shell sounds [23]. Some work has also been done to reduce their memory footprint [112] and computational costs [139, 16]. However, despite these advances, we have not witnessed wide spread adoption of such techniques in virtual environment applications, such as video games and training simulations. Pre-recorded samples and event-based playback still appear to be the most popular solutions. We believe that a major reason for this is that existing rigid body sound models still cost too much in terms of online memory requirements to make them practical without

compromising quality.

Virtual environments in the near future, such as video games and training simulations, will likely contain a large number of objects that are simulated using rigid body dynamics. For example, a kitchen may contain dozens of unique pieces of silverware, bowls, and plates of various shapes and sizes. Currently, storing the modal matrices necessary for sound synthesis for so many objects could potentially require dozens of gigabytes. This is impractical even with 64-bit machines, as such applications can only dedicate a limited amount of memory to sound synthesis. We suspect that this is a major reason why linear modal synthesis has not been adopted for such applications. In this paper, we propose a simple yet effective pipeline for compressing modal sound models, and we demonstrate its ability to achieve high compression ratios, allowing a hundred times more unique objects than previously possible, with a variety of objects while maintaining low approximation errors.

Our compression pipeline for the modal matrix is fundamentally based on down-sampling the mesh. Each row of the matrix represents a degree of freedom in the original finite element mesh, and its values dictate how the modes respond to an excitation of that degree of freedom. For high-resolution meshes, the dimensionality of the system can be in the hundreds of thousands, making this matrix very large. However, by simplifying the mesh using quadric error metrics [62], we are able to drastically reduce the number of rows in this matrix. We present this method in Section 4.4. Furthermore, we also explored methods specifically for objects which exhibit symmetry. Man-made objects are often symmetric, such as the cylindrical symmetry present in wine glasses, bowls, plates, and bells, and this leads to symmetry in their modes as well. In Section 4.5, we show that high compression ratios can be achieved with very low errors for such objects.

Please refer to Section 3.3.3 for a review of the modal sound synthesis pipeline. We will use the same notation in this chapter, referring to the modal basis matrix as **U**. The modal matrix can come from an eigenanalysis of a volumetric deformation model [103] or a thin shell model (Section 3.3.1). Both have the same memory-cost issues, and both can be addressed using the compression methods proposed in this chapter.

4.2 Related Work

4.2.1 Sound Model Compression

Raghuvanshi and Lin [112] address the memory problem by aggregating modes of similar frequency. The rationale is that such modes can be essentially treated as a single mode, and so columns of the basis matrix are simply summed together if their frequencies differ by a small threshold amount. They achieve space reduction by a factor of about 15. However, we not only achieve much higher compression, we also do not fundamentally lose important beating effects that are very audible for important objects such as bells.

Richmond [117] addresses adaptive sampling for the purposes of recording sound samples from real-world examples. The "acoustic distance" of two samples was measured as a function of their frequency content. If the acoustic distance of two samples was sufficiently large, another sample would be taken in between. While similar to our problem, this only addresses the distance between two recorded sounds. We are inter-

ested in the distance between modal values that are many layers removed from the actual synthesized sound.

Lastly, van den Doel et al. [139] uses knowledge of human perception to cull modes at run-time depending on which modes are active. Of course, this does not address the run-time memory costs, as all modes must still be available online in case they are not culled.

4.2.2 Mesh and Field Compression

The field of mesh compression also tackles similar problems of compressing data defined over a surface. One approach is to project the vertex positions onto a basis that is only dependent on the vertex connectivity [74]. Such a basis can be created using the eigenvectors of the mesh's Laplacian matrix. One can imagine using this to compress vibration modes, but this is ill-suited for run-time reconstruction, as the Laplacian eigenvectors must be computed. If they are stored, then this would reduce to approximating one set of eigen-modes with another, and would likely result in poor compression. Second-generation wavelets [86, 120] offer another approach to compressing data on arbitrary surfaces. However, wavelets do not perform well on smooth, periodic functions, like vibration modes.

Other research in computer graphics address similar problems, even though the intended application is not sound synthesis. Precomputed radiance transfer [128] has a memory requirement similar to modal sound synthesis. At run-time, a vector (or matrix) of spherical harmonic coefficients must be known for each point on the surface. To address this,

a clustered PCA process can be used to compress these coefficient vectors into a lower-dimensional approximation [127]. One can imagine applying this approach to vibration modes, treating each r-vector of modal responses like the coefficient vectors. However, this is unlikely to work well, since our modal responses are not particularly correlated from one mode to another. Later, [129] made PRT practical for local effects on deforming surfaces by using a subset of the spherical harmonic basis, the zonal harmonics. They fit zonal harmonics to the precomputed transfer data by optimizing an orientation axis and a weight for a number of basis function. We attempted to use a similar nonlinear fitting approach to approximate modes, but due to the geometric complexity of modal data, this did not yield satisfactory results. Lastly, recent work by Seo et al. [123] addresses blendshape compression for animation, and in many ways the problem is similar to ours. However, their compression largely relies on the locality of most blendshapes, such as the motion of an eye lid in facial animation. Vibrational modes, on the other hand, tend to have more global effect.

4.2.3 Symmetry

There exists a large body of work in computer graphics on detecting symmetries and making use of it. We refer the reader to a survey by Mitra et al. [96] for a comprehensive overview of research in symmetry analysis and applications. One of the major applications of symmetry research is indeed compression, and many researchers have proposed different methods for analyzing and representing symmetric objects [82, 95]. Our approach builds upon the work of Martinet et al. [89], but many other techniques are complementary and may provide even further compression. Our contribution is mainly in identifying which types of symmetry-exploiting compression are particularly effec-

tive for the displacement fields of rigid body sound models.

4.3 Problem Statement

We first state the problem and the objective metrics we will use to judge the quality of a compression scheme. In order to use a rigid body sound model at run-time, one must have access to the displacement fields $d_m(x)$ of all modes $m \in [0, r)$. They essentially encode information about how each mode would respond to external force vectors f experienced by the object. We will assume that these objects and models will only be used in rigid body dynamics scenarios where all external forces are surface impacts. Thus, we are only concerned with displacement fields on the surface S of the object. When an impact force f is experienced at point $x \in S$, the effective force on mode m is given as:

$$f_m = d_m(x)^T \ f \tag{4.1}$$

The displacement field $d_m(x)$ for a mode m is represented by the surface mesh S of the object and the eigen vector u_m, the m-th column of the basis matrix $\mathbf{U}$. The eigenvector stores values of $d_m(x)$ for vertices of the mesh $x \in \texttt{verts}(S)$, so if $N_{verts} = |\texttt{verts}(S)|$, then $u_m \in \mathbb{R}^{3N_{verts}}$ and $\mathbf{U} \in \mathbb{R}^{3N_{verts} \times r}$. For any point that is not a vertex, we find the triangle that contains it and linearly interpolate:

$$d_m(x) = \sum_{i=1}^{3} w_i d_m(v_i) \tag{4.2}$$

Here, v_i are the vertices of the triangle and w_i are the barycentric coordinates of x in that triangle.

Mode Approximation Error: In order to quantify the performance of a mode approximation, or a compressed mode, we must define an error metric. We choose a simple definition that basically measures how well the original eigenvector u_m can be approximated by the representation. If $d(x)$ is the displacement field's ground truth and $\tilde{d}(x)$ is the approximation, the error ε is defined as:

$$\varepsilon = \frac{\sqrt{\sum_{x \in \texttt{verts}(S)} \left| d(x) - \tilde{d}(x) \right|_2^2}}{|u_m|_2} \tag{4.3}$$

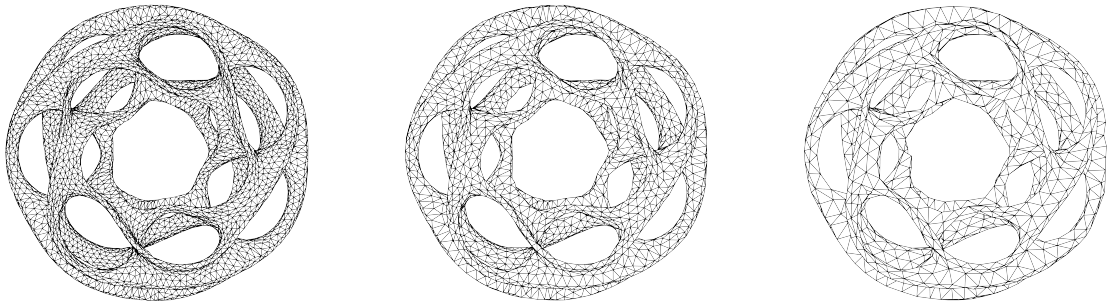
Again, this can be viewed simply as the relative error of the u_m approximation:

$$\varepsilon = \frac{|\tilde{u}_m - u_m|_2}{|u_m|_2} \tag{4.4}$$

Where $\tilde{u}_m$ is the vector of stacked $\tilde{d}(x)$ values for $x \in \texttt{verts}(S)$.

4.4 Mesh Simplification

Our approach to compressing the modal data for general objects is to downsample the original surface mesh S in a guided manner so that we only need to store displacement values for vertices of a downsampled mesh S_*. Figure 4.1 shows three such meshes for the heptoroid model. At run-time, we will represent the displacement field $d(x)$ of a given mode by linearly interpolating the values stored on vertices of S_*, just as before, but since S_* will have less vertices than S, there will be far fewer values to store. To perform the downsampling, we build on the quadric error simplification methods of [47, 61, 62].



8189 Vertices 4095 Vertices 1638 Vertices

Figure 4.1: **Simplified meshes for a heptoroid**

4.4.1 Quadric Error Simplification

We first give a brief review of quadric error mesh simplification as presented by [47]. The motivation is to simplify meshes using a sequence of greedy edge collapses in a way that minimizes some notion of geometric approximation error. A naive approach would involve evaluating global geometric error for every potential collapse, and then proceeding with the collapse that introduces the least amount of global error. However, this can be prohibitively inefficient even for meshes of moderate resolution. Instead, a quadric error metric is used that locally approximates error. When a collapse is done, only a small neighborhood of other collapses need to recompute their quadric errors. The whole greedy algorithm is then $O(N_{verts})$, compared to the $O(N_{verts}^2)$ or worse naive approach.

The geometric quadric error metric $Q^f(v)$ measures a vertex v's perpendicular distance from a given face f. However, beyond geometric distance, attribute interpolation error can also be minimized [61]. For example, if the surface has a texture map defined as colors on its vertices, one would want to simplify the mesh in a way that preserves the details of the colors as well as the geometry. If each vertex has an attribute vector

$s \in \mathbb{R}^M$, the full quadric error metric becomes:

$$Q^f(v = (p,s)) = Q_p^f(v) + \lambda \sum_{j=1}^{M} Q_{s_j}^f(v) \tag{4.5}$$

Where $Q_p^f(v)$ is the geometric error for position p, and $Q_{s_j}^f(v)$ measures linear interpolation error for attribute s_j. The parameter λ weighs the importance of geometry error versus attribute error, but we found that setting it to 1.0 with some normalization (discussed in the next section) was sufficient for all our examples. Then, the specifics of our problem allow us to use this simplification algorithm directly. Our modes are usually very smooth functions and do not exhibit sharp discontinuities, allowing us to ignore the wedge-based formulation presented in [61]. Furthermore, because our surfaces are necessarily manifold (they are surfaces of volumetric tetrahedral meshes), there is no need for boundary preservation. We also do not try to preserve geometric volume, although it is possible this would further improve performance. Lastly, because we may have hundreds of modes, the algorithm of [61] can be inefficient as it is $O(r^2)$. To make the algorithm $O(r)$, we implement the improved quadric optimization solve presented in [62].

4.4.2 Simplification for Modes

Simplification Details: For our problem, we set the attribute vector s for each vertex v as its corresponding 3-by-r block-row of the basis matrix. We stack it as a single $3r$-vector, so $s = \in \mathbb{R}^{3r}$ and $M = 3r$. As mentioned previously, we normalize the attribute vectors s by multiplying them by a factor α. This is to make sure the geometry vs. attribute importance parameter λ can be kept constant independent of the size of the

mesh or the magnitude of the displacement vectors. The factor is computed as:

$$\alpha = \frac{\max_v |v - v_c|_2}{\max_{m,v} |d_m(v)|_2} \tag{4.6}$$

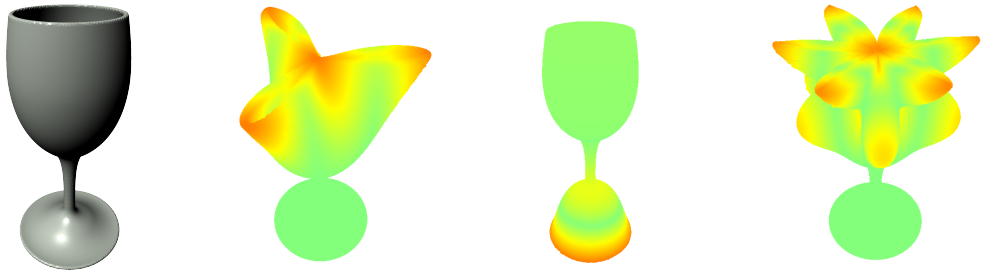
Here $v \in \texttt{verts}(S)$ and v_c is the centroid of the mesh. The numerator is the radius of the mesh, so the size of the mesh does not affect relative importance, and the denominator does the same for the size (or maximum magnitude) of the displacement field.

Usage: The simplification process yields a sequence of simplified meshes S_i, each with fewer vertices than the previous, which attempt to balance geometric approximation error with displacement field approximation error for all modes. For each mesh S_i, we compute mode approximation errors $\varepsilon_{i,m}$ for all modes m using Equation 4.3. The displacement field approximation $\tilde{d}_m(x)$ is computed using Equation 4.2, except the surface is S_i and the basis matrix is much smaller. If S_i has N_i many vertices, then our compressed basis matrix will only contain $3N_i$ rows, resulting in a compression ratio of $1 - N_i/N_{max}$. To choose a mesh S_* to use at run-time, the user specifies a maximum mode approximation error ε_{max}. We then choose $S_* = S_i$ such that $\forall m \in [1, r]\ \varepsilon_{i,m} \leq \varepsilon_{max}$.

Lastly, please see the Appendix A for a description of "perceptual weights," which was an attempt to more intelligently guide the simplification process by exploiting auditory perception and various other factors. It did not work out, but the exposition is included for completeness.

4.5 Exploiting Symmetry

Many man-made objects exhibit symmetry, and this results in modes that are also highly symmetric. Figure 4.2 shows a wine glass model along with a few of its modes. The surface clearly exhibits cylindrical symmetry along the vertical axis, and we see that its modes also exhibit such symmetry, although in different ways. This is clearly an opportunity for compression, and in this section we present our pipeline for automatically detecting such symmetries and then utilizing that information for compression.



Surface Geometry Mode 8 (1.6 khz) Mode 16 (3.2 khz) Mode 67 (12.9 khz)

Figure 4.2: **Selected wine glass modes:** *The top-left is the original wine geometry, and 3 typical modes are shown. The colors correspond to the magnitude of the displace field over the surface.*

Overview: We first present a method of automatically detecting cylindrical symmetry from Marinet et al. [89]. Then, given the best axis of symmetry, we compress the modes in two steps:

- Many modes exhibit either continuous cylindrical symmetry themselves, such as mode 16 of Figure 4.2, or they exhibit discrete rotational symmetry of some order, as with modes 8 (3-fold) and 67 (6-fold). This can clearly be exploited to compress an individual mode: If a mode has n-fold rotational symmetry, we can

simply store one "slice" of the data and compress it down to $1/n$ of its original size. If it is continuously symmetric, we can just choose some large value of n, such as 10.

- Modes often come in pairs that are actual congruent to each other. Figure 4.7 shows some examples of mode pairs which are obviously rotations of each other, even if in some cases they do not individually exhibit rotational symmetry (as with modes 1 and 2). This can also be exploited by detecting the rotation which will rotate one mode into the other and only storing one of them for a 50% compression for that pair.

4.5.1 Symmetry Detection

We provide a brief summary of the method presented in Martinet et al. [89] for automatically detecting cylindrical symmetries in geometry. Their method is more general and can detect other types of symmetry, but for our purposes we will focus on cylindrical symmetries. Given a 3D surface S *centered at the origin*, the goal is to detect a direction α around which the surface can be rotated by any angle ϕ and still remain nearly-identical to the original geometry:

$$R(\alpha,\phi) = R_{\alpha}^{-1} R_{\phi} R_{\alpha} \tag{4.7}$$

$$\forall \phi \in [0, 2*\pi),\ R(\alpha,\phi) S = S \tag{4.8}$$

Here, R_{α} is the rotation which maps direction α to the positive z-axis and R_{ϕ} is a rotation about $+z$ by ϕ-radians (we adopt the convention where ϕ is the azimuthal angle and θ is the polar angle).

Generalized Moment Functions: Finding the axis α is very expensive if done in a brute force manner. To do this efficiently, [89] proposes a *generalized moment function* of order p, $\mathcal{M}^{2p}(\omega)$, such that any symmetries present in S must also be present in $\mathcal{M}$ (Figure 4.3 shows examples of surfaces and their corresponding $\mathcal{M}$ functions). So if a cylindrical symmetry exists about α, then:

$$\mathcal{M}^{2p}(R(\alpha,\phi)\omega) = \mathcal{M}^{2p}(\omega) \tag{4.9}$$

The converse is not true, so a symmetry in $\mathcal{M}$ may not necessarily exist in S. However, symmetries *not* in $\mathcal{M}$ definitely do not exist in S either, so we can use $\mathcal{M}$ to efficiently prune the search space. To do so, we need to efficiently evaluate $\mathcal{M}$. Its actual definition is a surface integral over S, which is expensive, but [89] shows that it can be expressed in terms of real-valued spherical harmonics [60] with pre-computed coefficients:

$$\mathcal{M}^{2p}(\omega) = \sum_{l=0}^{p} \sum_{m=-2l}^{2l} C_{2l,m}^{2p} Y_{2l}^{m}(\omega) \tag{4.10}$$

Y_{2l}^{m} are the real-valued spherical harmonics and C are the coefficients. Please refer to [89] for the derivation and formula for the coefficients. The upshot is that they can be computed once for efficient evaluation of $\mathcal{M}$.

Finding the Axis of Symmetry: Given $\mathcal{M}$, we now wish to find an axis of cylindrical symmetry. To do so, we utilize the following property of spherical harmonics expansions: the expansion is cylindrically symmetric about the z-axis if and only if it decomposes into nothing but zonal harmonics ($m = 0$). This can be checked, to a tolerance, by summing the magnitudes of non-zonal coefficients:

$$\mathcal{Z} = \sum_{l=0}^{p} \left(\sum_{m \in [-2l,2l]-\{0\}} |C_l^m| \right) \tag{4.11}$$

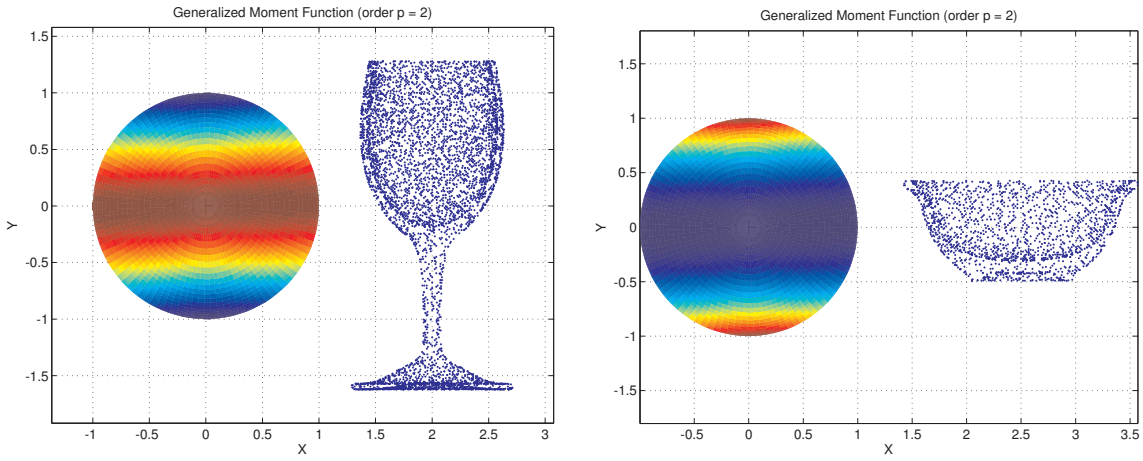


Figure 4.3: **Generalized Moment Functions** *for the wine glass (left) and the plastic bowl (right). For our examples, we found that using order $p = 2$ was sufficient to detect existing symmetries. It is clear from both plots that symmetries in the models (the blue point clouds) are also present in the directional $\mathcal{M}^{2p}$ functions.*

If $\mathcal{M}$ has cylindrical symmetry about the axis α, then $\mathcal{Z}$ should be zero after we rotate the spherical harmonics coefficients by R_α^{-1}:

$$\mathcal{Z}(\alpha) = \sum_{l=0}^{p} \left(\sum_{m \in [-2l,2l]-\{0\}} |D_l^m| \right) \tag{4.12}$$

Here, D_l^m are the coefficients rotated by R_α^{-1}, and we use the method of [66] to efficiently compute them. Figure 4.4 shows the $\mathcal{Z}$ function evaluated for two models. Our goal is to find the direction α that minimizes it, and we do so by subdividing an icosahedron as in [89]. We first refine the icosahedron three times using Loop subdivision [85] and evaluate $\mathcal{Z}(\alpha)$ for all vertices of the subdivided geometry. We then sort all vertices by their $\mathcal{Z}(\alpha)$ values, subdivide the bottom 10%, and evaluate $\mathcal{Z}(\alpha)$ for all new vertices. We repeat this 2 more times and return the vertex (which is a point on a unit sphere, and thus a direction) with the lowest $\mathcal{Z}(\alpha)$ value. This gives us our best guess for the axis of cylindrical symmetry, if any exists at all. As for the order of the generalized moment functions, we found that using $p = 2$ was sufficient for our examples. The next two parts of the pipeline will check and utilize this information.

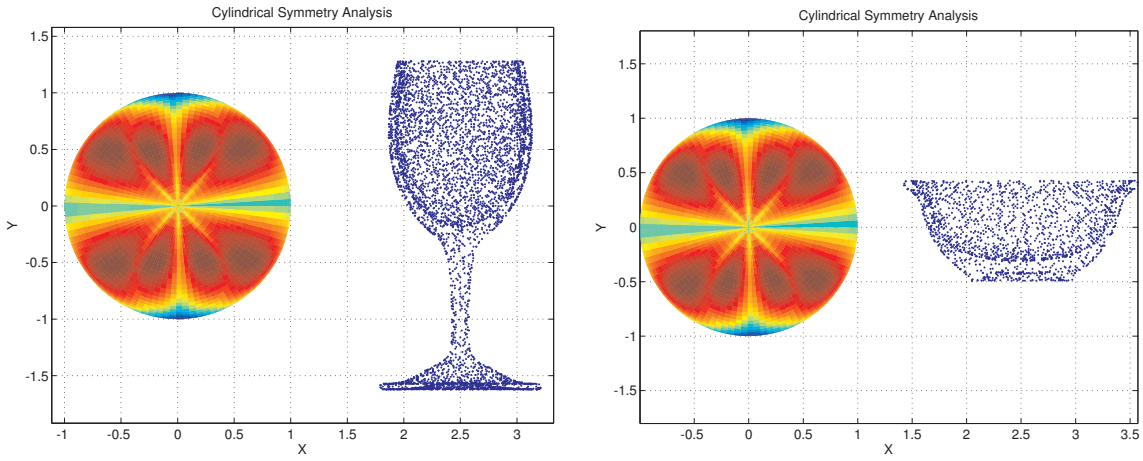


Figure 4.4: **Cylindrical Symmetry Analysis** *for the wine glass (left) and the plastic bowl (right). The spheres visualize evaluations of $\mathcal{Z}(\alpha)$ for values of α sampled over the unit sphere, using Equation 4.12. The color scheme maps dark blue to the lowest value and bright red to the highest. Notice that in both cases, the analysis correctly concludes that the lowest values occur at the +/- Y directions, which is indeed the axis of cylindrical symmetry for both objects.*

4.5.2 Rotational Symmetry

If the surface is cylindrically symmetric about direction α, or nearly so, its modes will often exhibit discrete or continuous rotational symmetry within themselves along the same direction. Let $d(x)$ be the modal displacement vector for a single mode at surface point x. If the mode exhibits n-fold rotational symmetry about α, this means that $\forall x \in S$:

$$R_x = R(\alpha,\, \phi_n(R_\alpha\, x)) \tag{4.13}$$

$$d(x) = R_x^{-1}\, d(R_x\, x) \tag{4.14}$$

Here, $\phi_n(x)$ returns the azimuthal angle of point x modulo the n-fold rotational symmetry. So if $\phi(x)$ is the actual azimuthal angle (assumed to be positive), then:

$$\phi_n(x) = \phi(x) - \lfloor \frac{\phi(x)}{\pi/n} \rfloor \pi/n \tag{4.15}$$

Thus, R_x is the rotation which maps any point x to its corresponding point in the first "slice" of the surface, given that the surface is made up of n congruent slices about α.

Figure 4.5 illustrates this.

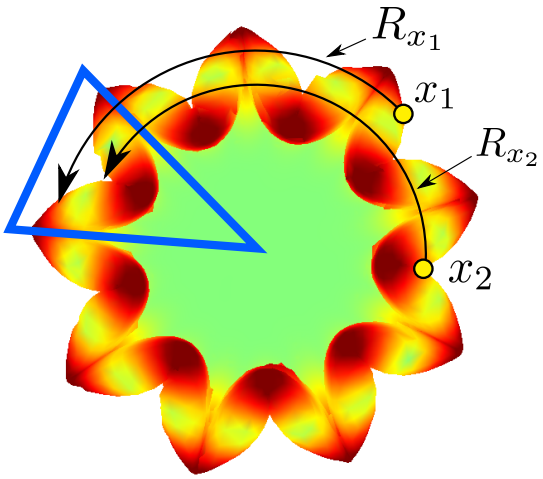


Figure 4.5: **Exploiting n-fold rotational symmetry** *for a single mode. Here we have a 9-fold rotationally symmetric mode from the plastic bowl model. We can trivially compress this to 1/9 of its original size by only storing the displacement samples within the "slice" highlighted in blue. For points outside this slice, we simply rotate them into their symmetrically equivalent position at run-time.*

Determining order of symmetry: Given the best guess for α from the previous section, we can now try to determine, for each individual mode, what the order n of rotational symmetry is. We do this by a simple brute force search over $n \in [2, n_{max}]$, computing the mode approximation error $\varepsilon_{m,n}$ for each mode m and order n with Equation 4.14 as the approximation. We found this to be efficient using closest-point datastructures [1] and $n_{max} = 10$. Then given an error tolerance ε_{max}, we select the highest order n_m, which will result in the highest compression, that meets the tolerance:

$$n_m = \max\{n : \varepsilon_{m,n} < \varepsilon_{max}\} \tag{4.16}$$

Order of symmetry behavior: Our initial hypothesis from looking at the modes is that order of symmetry would grow monotically with frequency. In the ideal case, each

pair of modes would exhibit a higher order of symmetry than the previous pair. The first one would be 2-fold, so we could just store $1/2$ of its data. The next one would be 3-fold, so we could store $1/3$, and so on. Then in theory, the size of our compressed approximation would be:

$$2\left(1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{N/2}\right)=2\sum_{k=1}^{N/2}\frac{1}{k}=2H_{N/2} \tag{4.17}$$

Where H_i is the i-th harmonic number. It is known to be $O(\log(n))$, which means our compression would improve the more modes we have. Unfortunately, the orders of symmetry do not increase monotonically, as is illustrated in Figure 4.6. There is clearly some structure, but it is more complex and the lower bound on the order of symmetry never seems to increase for higher-frequency modes.

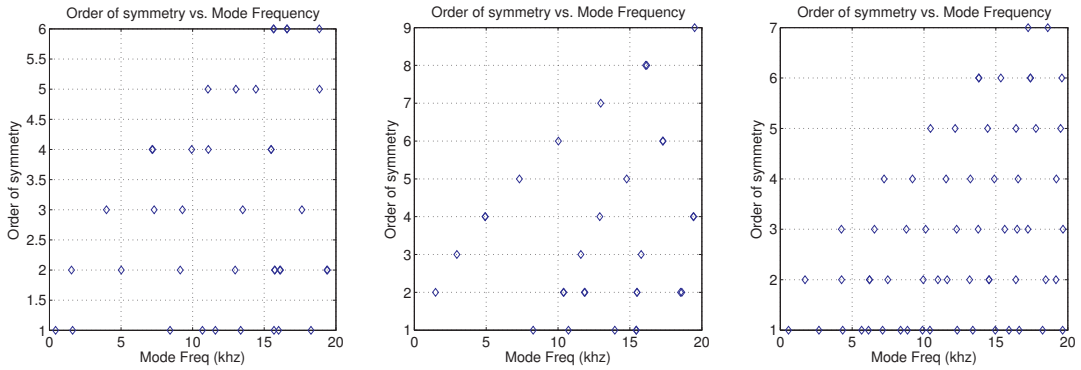


Figure 4.6: **Rotational Symmetry Order vs. Mode Frequency Plots** *for the three objects: wine glass (left), plastic bowl (middle), and bronze bell (right). It is clear that there is a non-monotonic relationship between order and frequency, but there does appear to be bands of monotonicity for subsets of modes. This is especially apparent for the plastic bowl (middle). The orders were determined using Equation 4.16 with* $\varepsilon_{max}=0.5$.

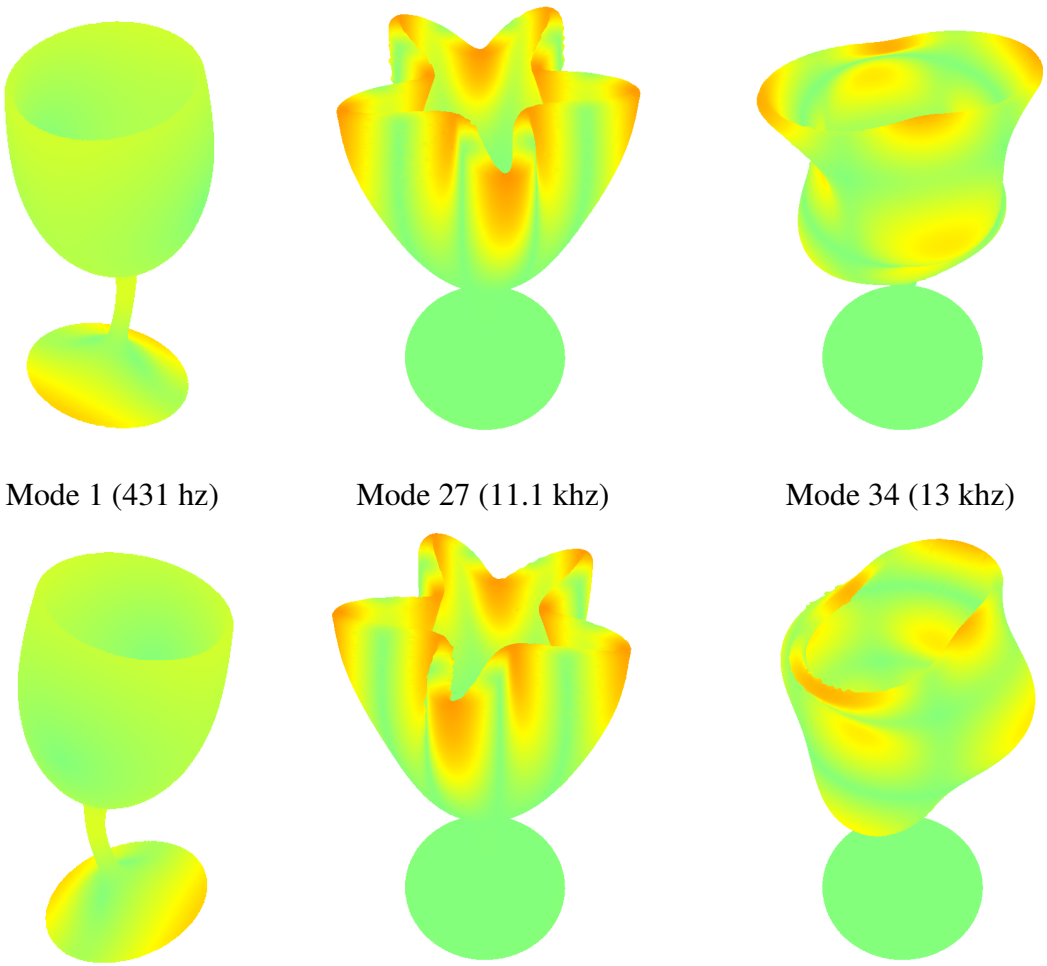


Figure 4.7: **Congruent pairs of wine glass modes:** *Each column is a pair of modes which are rotationally congruent with each other. We can immediately get 50% compression by simply discarding one of the modes and storing a rotation angle. Note that all pairs have very similar frequencies.*

4.5.3 Congruent Pairs

Beyond rotational symmetry within a single mode, an interesting charateristic of the modes as a whole is that they often come in rotationally congruent pairs. Figure 4.7 shows several pairs of modes which exhibit this. If we can detect such congruent pairs (a,b), we can completely toss out mode a and just store the rotation $\phi_{a,b}$ that maps one

to the other. We can then simply reconstruct the displacement field of a as:

$$R_{a,b} = R(\alpha,\, \phi_{a,b}) \tag{4.18}$$

$$d_a(x) = R_{a,b}^{-1}\, d_b(R_{a,b}\, x) \tag{4.19}$$

The overall approach of our algorithm is to first compute some guesses for congruent pairs and corresponding rotations and then check them with a more expensive operation. Much like the approach taken by [89], we summarize the angular structure of the modes in a low-dimension Fourier basis to prune the search space, and then we do a more rigorous verification of the candidates. Furthermore, we observe that congruent pairs are usually close to each other in frequency, so instead of doing this for all pairs (a,b), we only do it for pairs such that $a < b$ and $|a-b| \leq 2$ (assuming modes are numbered in order of increasing frequency).

For a given pair of modes (a,b), we first focus on the problem of finding a best rotation $\phi_{a,b}$. We compute a Fourier expansion that captures the angular variation of the displacement fields about α:

$$c_{a,m} = \int_S |d_a(x)| e^{im\phi(x)} dx \tag{4.20}$$

$$\mathcal{F}_a(\phi) = \sum_{m=0}^{N-1} c_{a,m} e^{im\phi} \tag{4.21}$$

Figure 4.8 shows examples of modes and their $\mathcal{F}(\phi)$ evaluations. Much like the general moment function $\mathcal{M}$, this representation allows us to efficiently search for possible rotations that map d_a to d_b. If a rotation by angle $\phi_{a,b}$ would accomplish this, then it must be that:

$$\mathcal{F}_a(\phi) = \mathcal{F}_b(\phi + \phi_{a,b}) \tag{4.22}$$

$$\mathcal{F}_a(\phi - \phi_{a,b}) = \mathcal{F}_b(\phi) \tag{4.23}$$

We also know that Fourier expansion coefficients can be rotated as follows:

$$\hat{c}_{a,m} = c_{a,m}e^{im\phi_{a,b}} \tag{4.24}$$

$$\mathcal{F}_a(\phi - \phi_{a,b}) = \sum_{m=0}^{N-1} \hat{c}_{a,m}e^{im\phi} \tag{4.25}$$

Then it follows from the uniqueness of Fourier expansions that any rotation $\phi_{a,b}$ that rotates $\mathcal{F}_a$ into $\mathcal{F}_b$ would also rotate the coefficients to make them equal, so it would minimize the objective function:

$$\Phi = \sum_{m=0}^{N-1} |\hat{c}_{a,m} - c_{b,m}|^2 \tag{4.26}$$

Thus, our goal is to find values of $\phi_{a,b}$ such that $\Phi \approx 0$. However, Φ will likely contain many local minima, so we need to make sure we sample the space of azimuth angles thoroughly. We do this by subdividing the search space $[0, 2\pi)$ into 20 equal bins and computing the local minima within each bin (using Matlab's *fminbnd* function). We output all minima $\phi_{a,b}^{(i)}$ for $i = 1 \ldots 20$ as possible rotations to be verified for all considered pairs (a,b).

For verification, we again do a brute force computation of mode approximation errors $\varepsilon_{a,b,i}$ using Equation 4.19 as the approximation for rotation candidates $\phi_{a,b}^{(i)}$. For each pair, we choose the rotation that yields the least error:

$$\phi_{a,b} = \phi_{a,b}^{(j)} : j = \mathbf{arg\,min}_i \, \varepsilon_{a,b,i} \tag{4.27}$$

Then given an error tolerance ε_{max}, we can toss out information for d_a if there exists a reference mode b such that $\phi_{a,b} < \varepsilon_{max}$.

Objects with Discrete Rotational Symmetry: For surfaces with discrete rotational symmetry, such as an extruded pentagon, we unfortunately do not observe rotationally

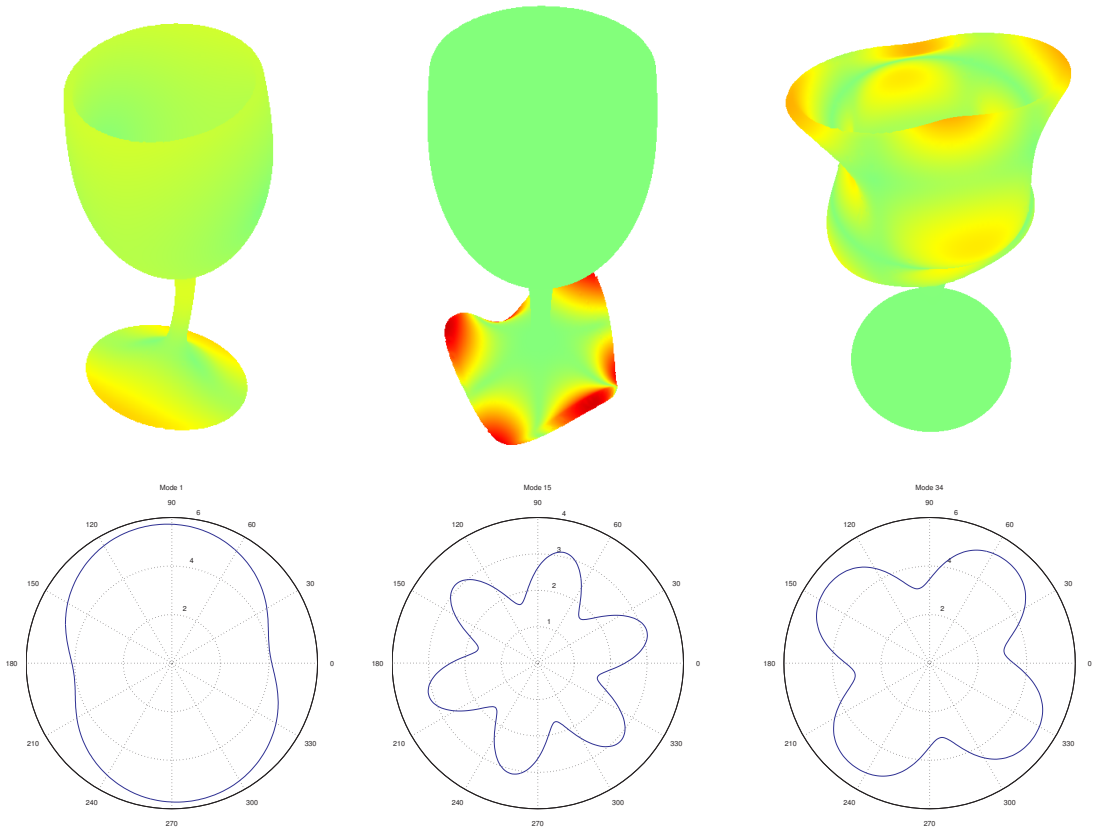
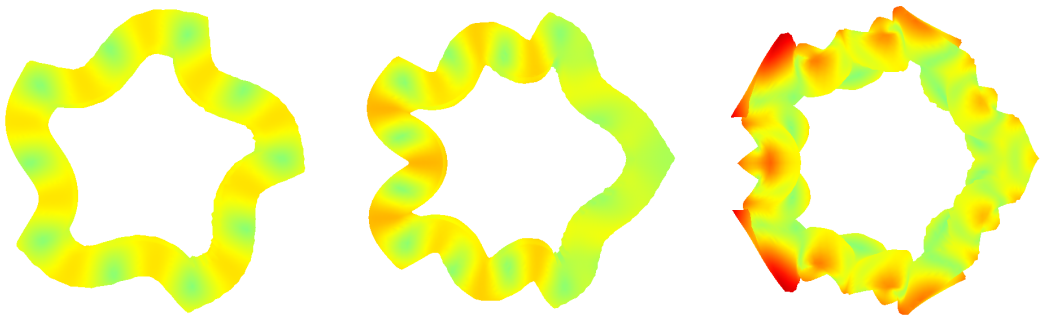
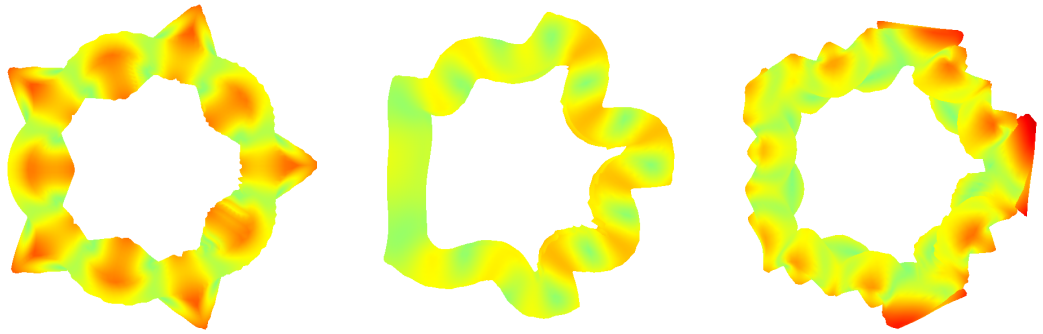


Figure 4.8: **Fourier expansions of mode angular variation:** *Each column has the visualized mode on the top and the evaluation of Equation 4.21 on the bottom. The expansion summarizes the angular-only variation of the displacement field, and this can be used to prune the search space when looking for rotationally congruent pairs.*

congruent pairs. Many individual modes still exhibit discrete rotational symmetry, but the pair-wise relationships between modes with similar frequencies are not as simple as in the continuously symmetric case. Pairs of modes that seem to complement each other do exist, but they are not aligned in a way that makes them congruent. See Figure 4.9 for examples of such pairs. Of course, as the order of rotational symmetry increases, shapes become more and more cylindrically symmetric, and thus approximating such modes as congruent pairs will result in lower and lower errors. We discuss this further in the results section.



Mode 20 (3.8 khz) Mode 39 (7.2 khz) Mode 73 (13.7 khz)



Mode 41 (7.5 khz) Mode 40 (7.2 khz) Mode 74 (13.7 khz)

Figure 4.9: **Modes of an extruded pentagon ring:** *We investigated the modes of a pentagonal shape, which has 5-fold discrete rotational symmetry. Unfortunately, unlike continuously symmetric shapes, such shapes do not exhibit the same kinds of compressible redundancy. Modes do exhibit 5-fold symmetry, such as the left column. However, congruent pairs are not apparent. The closest thing we found are pairs like those in the middle and right columns. They clearly complement each other, but due to the shape of the domain, they are not rotationally congruent.*

4.6 Results

We now present the results in the form of compression ratio vs. error tolerance plots for a number of different objects. The error tolerance corresponds to the error metric of Equation 4.3, and the compression ratio is defined as follows:

$$1 - \frac{\texttt{after}}{\texttt{before}} \tag{4.28}$$

So a higher compression ratio corresponds to more compression. We also present statistics for the objects and sound models themselves, as well as timings for various stages

Model	r	N_0	N_*	Orig. Size	Comp. Size	Comp. Ratio	Simp. Time	Error Eval Time
Wine Glass	74	50538	1517	229 MB	3.6 MB	97%	22m	16m
Armadillo	24	34387	516	22 MB	321 KB	98%	26m	7m
Letter A	34	7534	641	34 MB	549 KB	91%	3m	2m
Heptoroid	194	81884	4504	379 MB	21 MB	94%	52m	98m

Table 4.1: **Mesh simplification statistics and timings:** *The compressed models were chosen with $\varepsilon_{max} = 5\%$, and Figure 4.11 shows the actual simplified geometries along with the original surfaces. The sizes reported include both the mesh files (stored in the standard OBJ format) and the modal matrix (stored as binary-encoded double-precision floats). The simplification times are reported in minutes, and they were ran on an 8-core 2.66GHz Xeon X5355 machine with 8GB of RAM. The Error Eval Time (also in minutes) refers to the time taken to evaluate mode approximation errors $\varepsilon_{i,m}$ to obtain data for the plots in Figure 4.10.*

of the pipeline. We present results separately for mesh simplification and symmetry, but the two are complimentary for cylindrically symmetric objects.

4.6.1 Mesh Simplification

Here we present results using our mesh simplification pipeline for four models: a wine glass, a plastic armadillo, a plastic letter "A", and a bronze heptoroid. Figure 4.10 shows their compression ratio vs. error tolerance plots, Table 4.1 lists statistics and timings, and Figure 4.11 shows the actual simplified geometries. For an error tolerance of $\varepsilon_{max} = 5\%$, we were able to achieve compression ratios of 91-98% for all four examples.

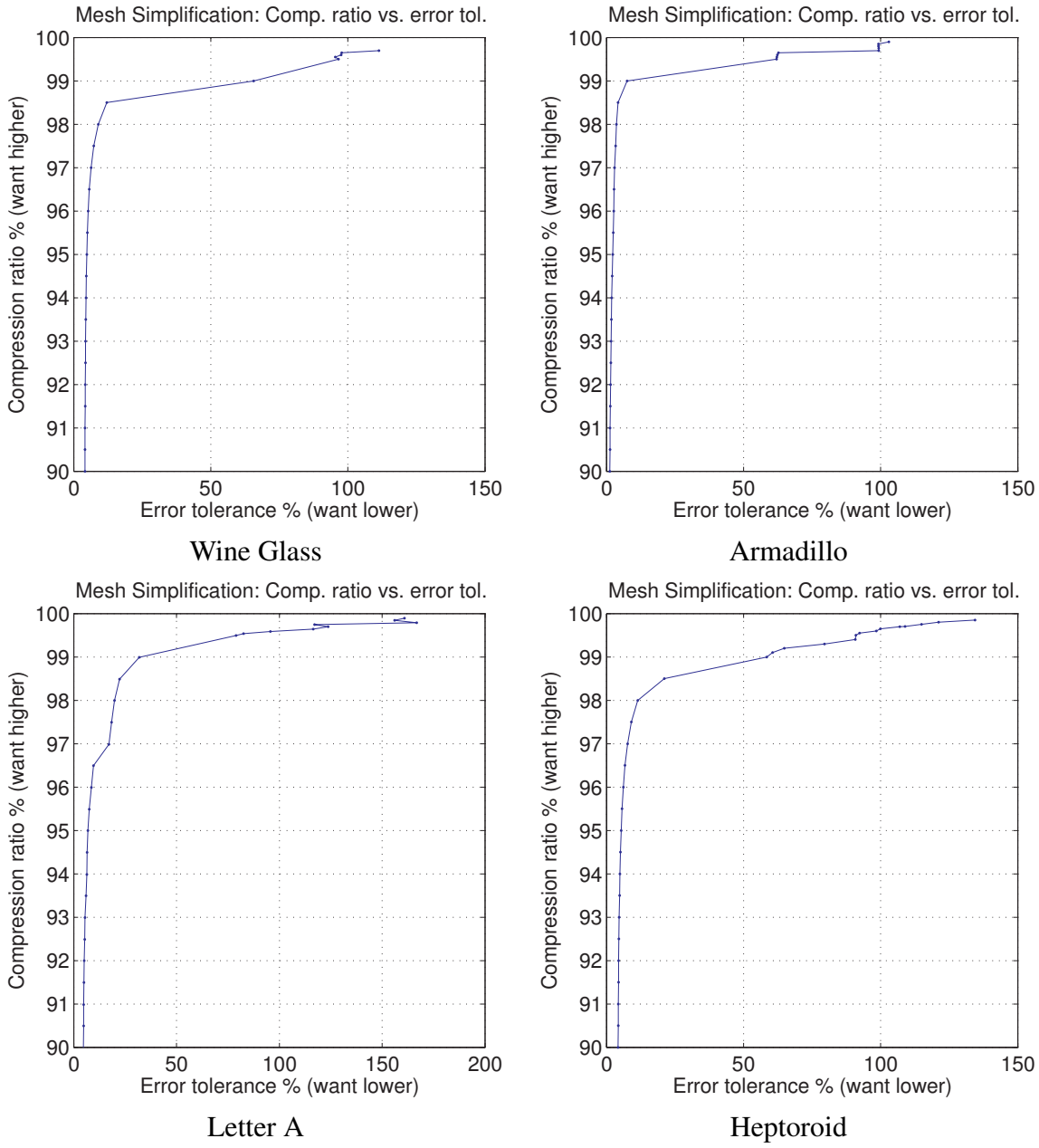


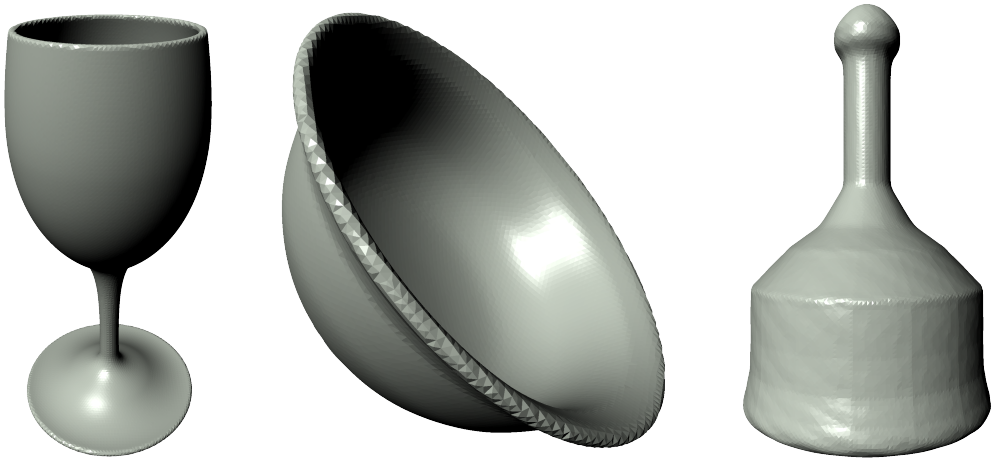
Figure 4.10: **Compression Ratio vs. Error Plots** *for the mesh simplification pipeline. Each data point corresponds to a simplified mesh S_i, the compression ratio is computed as N_i/N_0 where N_0 is the number of vertices of the original surface mesh, and the error is the maximum mode approximation error over all modes.* **NOTE:** *the Y axes begin at 0.9, since approximation error was already very low even with 90% compression. In all cases, we are able to achieve over 90% compression with only 10% maximum approximation error.*

Figure 4.11: **Simplified meshes for $\varepsilon_{max} = 5\%$**

4.6.2 Exploiting Symmetry

Here we present results using only our symmetry compression pipeline for three models: a wine glass, a plastic bowl, and a bronze bell. Figure 4.12 shows renderings of the three models, Table 4.2 lists some model statistics, Table 4.3 lists detailed timings for all steps of the pipeline, and Figures 4.13, 4.14, and 4.15 present ratio vs. error plots.

It should be noted that the compression ratios presented in the plots and tables are theoretical and were not derived from actual file sizes. If mode m was designated as n-fold symmetric, then it was assumed to be compressible down to $1/n$ of its original size. For cylindrically symmetric modes, we used $n = 10$. If two modes were designated as a congruent pair, then it was assumed one could be completely discarded. This does not take into account some amount of overhead, such as the angle $\phi_{a,b}$, but such overhead is likely to be insignificant compared to the modal data.

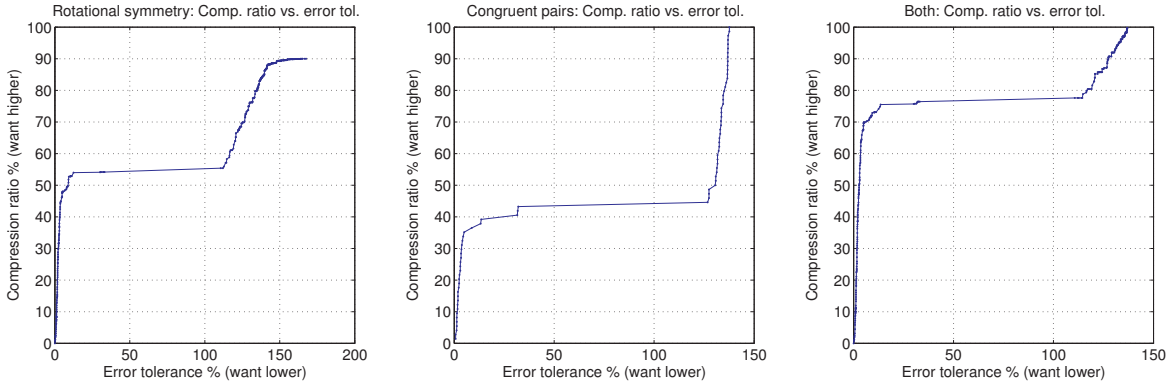


Wine Glass Plastic Bowl Bronze Bell

Figure 4.12: **Renderings of Cylindrically Symmetric Examples**

Model	r	# Verts	# Tris	# Cong. Pairs	Highest Sym. Order	Comp. Ratio
Wine Glass	74	50,538	101,072	27	6	73%
Plastic Bowl	49	18,370	36,736	18	9	77%
Bronze Bell	134	41,974	83,944	54	7	73%

Table 4.2: **Symmetry compression statistics:** *The detected symmetry statistics, namely the number of congruent pairs and the highest symmetry order, were chosen with* $\varepsilon_{max} = 10\%$. *As noted before, the compression ratio is a theoretical value based on reasonable assumptions.*

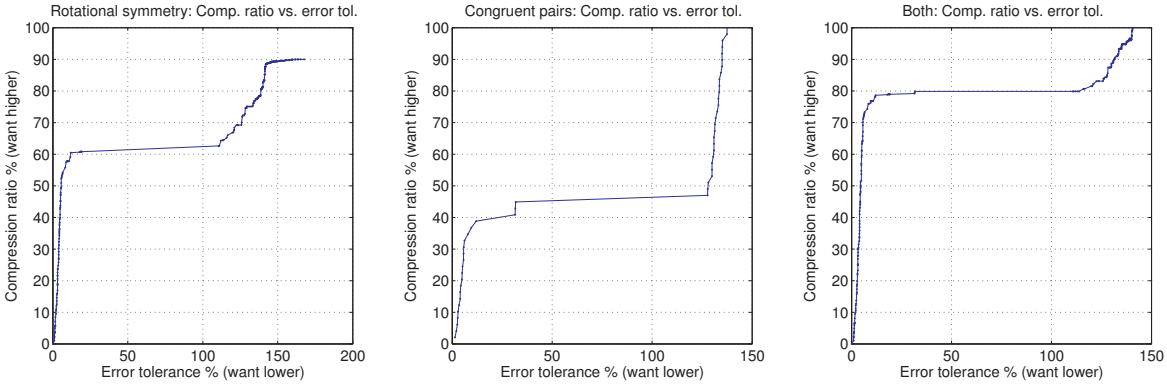


Rotational Symmetry Only Congruent Pairs Only Combined Compression

Figure 4.13: **Compression Ratio vs. Error Plots for Wine Glass**

Model	$\mathcal{M}^{2p}$ Coefs.	Axis (α)	$\varepsilon_{m,n}$	$\mathcal{F}$ Coefs.	$\phi_{a,b}^{(i)}$	$\varepsilon_{a,b,i}$
Wine Glass	01:33	00:03	20:05	04:35	00:09	89:26
Plastic Bowl	00:38	00:03	04:22	01:09	00:13	21:10
Bronze Bell	01:25	00:03	30:54	07:26	00:20	139:14

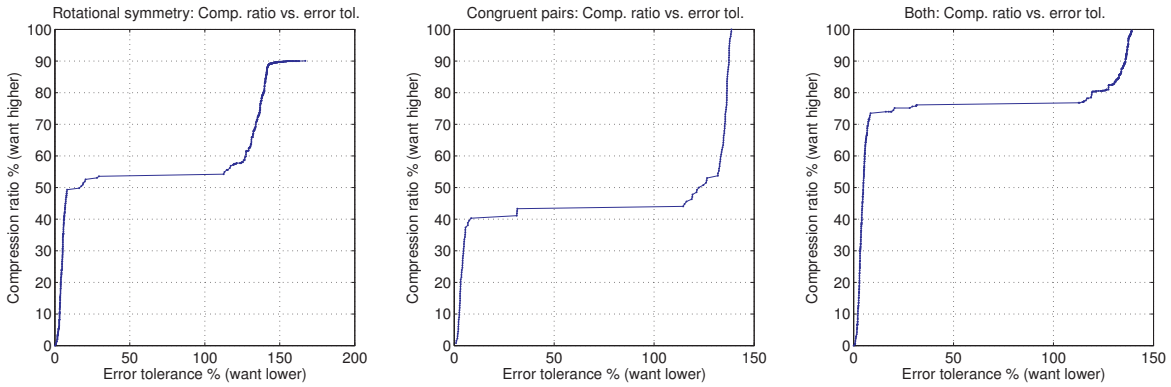
Table 4.3: **Symmetry compression pipeline timings:** *Each column corresponds to a step in the pipeline, and all timings are given in MM:SS format. From left to right, the columns correspond to the following parts of the pipeline: The $\mathcal{M}$ coefficients are computed using a surface integral, taking time proportional to the number of triangles. The axis of symmetry α is found by minimizing Equation 4.12. The errors $\varepsilon_{m,n}$ are computed to determine order of symmetry per-mode, and they are used in Equation 4.16. The coefficients for the $\mathcal{F}$ function are computed as in Equation 4.21. The candidate angles $\phi_{a,b}^{(i)}$ were computed using Matlab in a single threaded script, and their evaluated errors $\varepsilon_{a,b,i}$ are used in Equation 4.27. All pipeline steps were ran on an 8-core 2.66GHz Xeon X5355 machine with 8GB of RAM, and unless otherwise noted, reasonable effort was put into parallelization.*



Rotational Symmetry Only | Congruent Pairs Only | Combined Compression

Figure 4.14: **Compression Ratio vs. Error Plots for Plastic Bowl**

We also wanted to explore the effect of mesh resolution on the performance of the compression pipeline. As noted previously, objects that only exhibit discrete rotational symmetries do not produce modes with congruent pairs, and thus that part of the compression algorithm would not produce high compression with low errors. However, even if an object is supposed to be cylindrically symmetric in theory, it is never perfect due to mesh resolution errors. Thus, if an object is not well-tessellated, it may actually behave like an object with discrete rotational symmetry and thus not perform well in our



Rotational Symmetry Only | Congruent Pairs Only | Combined Compression

Figure 4.15: **Compression Ratio vs. Error Plots for Bronze Bell**

pipeline. We tested this by processing the bronze bell model at three different resolutions (see Figure 4.16) and plotted the compression vs. error curves in Figure 4.17. As expected, the curve does not rise as quickly for lower resolutions. Of course, lower resolutions are already cheaper to store to begin with.

4.7 Conclusion

We have demonstrated that some simple techniques can be used to dramatically reduce the memory costs of (nearly) rigid-body modal sound models. Mesh simplification, essentially downsampling the surface displacement field, and exploiting cylindrical symmetry when it exists both produce very favorable compression vs. error curves. Just using the simpliication alone we consistently achieve 95% or more compression ratios for 10% relative approximation error. Using only the symmetry-based techniques, we can achieve at least 70%. Furthermore, these two techniques are trivially complementary: the information that we do need to store for the symmetry-based techniques, the "slices" of the meshes, can be compressed further using simplification.

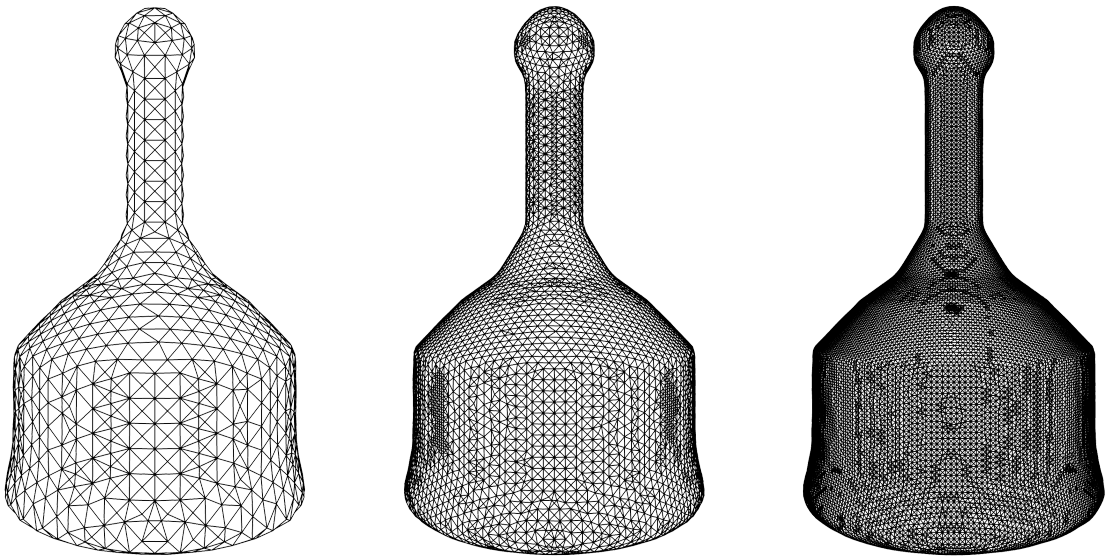


Figure 4.16: **Three meshings of the bell object** *were used to investigate the effect of meshing resolution on the effectiveness of rotational symmetry compression. The meshes were produced by varying the parameters of the isosurface stuffing algorithm presented in [79]. From left to right, the number of tetrahedra (number of surface vertices) for the meshes were: 13,065 (2,066) for Low, 90,991 (10,886) for Medium, and 390,263 (41,974) for High.*

Our goal is to make sound synthesis practical for the virtual environments of the future. In this paper, we have addressed the memory costs of modal sound models. In applications where memory is limited, such as real-time virtual environments, video games, and even large-scale offline rendering, using the methods presented can make modal sound synthesis much more practical. Of course, other bottlenecks, such as the integration of nonlinear deformation dynamics, still exist. Large models, such as the floor of a whole room or even whole buildings themselves, are still a challenge due to prohibitive pre-process costs during the parameter tuning cycle.

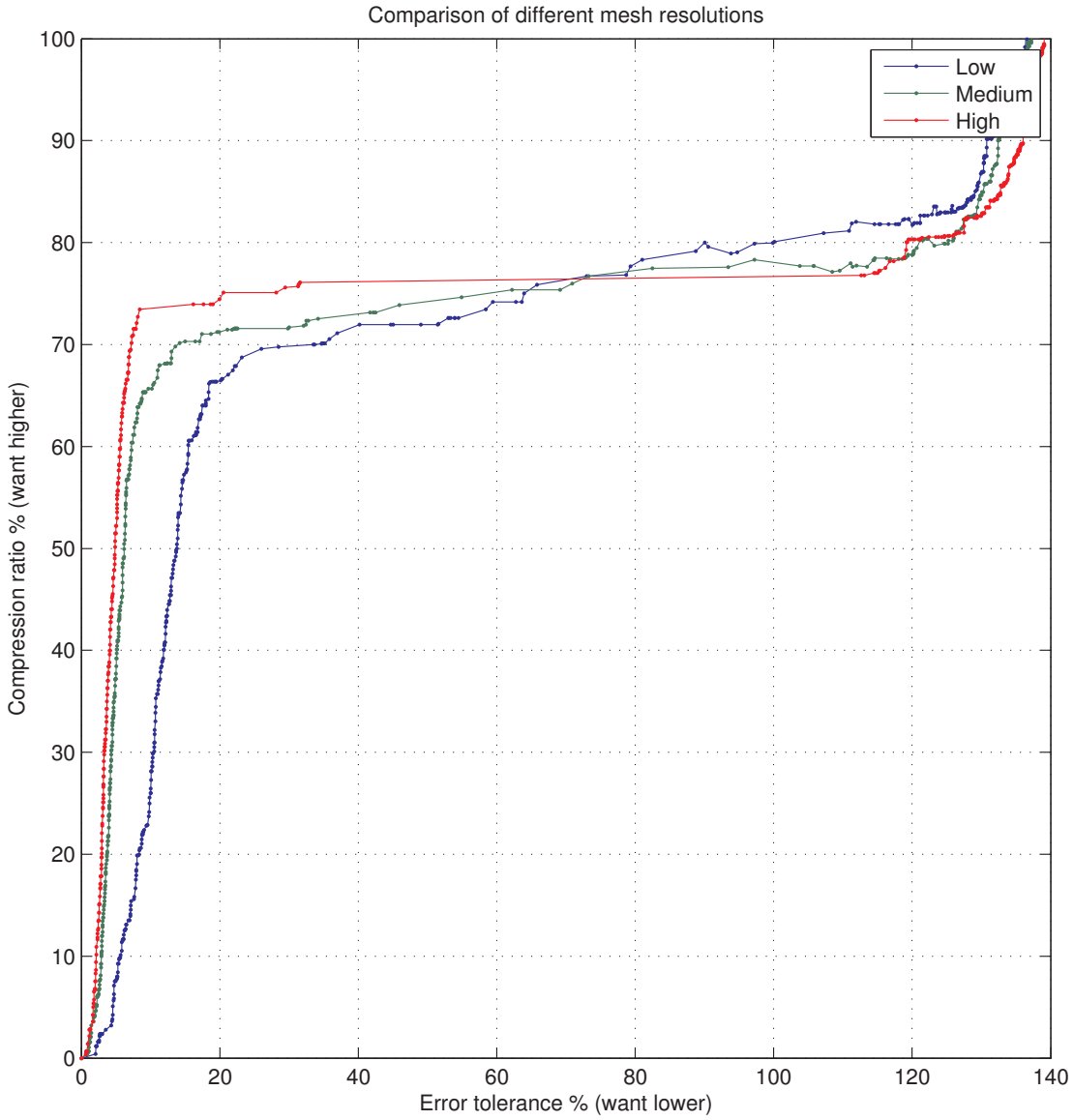


Figure 4.17: **Effect of mesh resolution on symmetry compression:** *Here we plot the compression ratio vs. error tolerance curves for the bronze bell at three different meshing resolutions. See Figure 4.16 for a visualization of the meshes and resolution statistics. The legend labels correspond to that figure's labels.*

4.7.1 Future Work

Exploiting Perceptual Effects: There is much literature from the psycho-acoustics community studying human perception of rigid-body sounds, both synthetic and recorded [49,

92]. One consistent result from these experiments is that sounds which are highly damped are less distinguishable from each other. This suggests that we can compress highly damped modes, typically the high frequency modes, much more than others without a noticeable difference in quality or perception. This could be accomplished by simply using fewer bits to encode a mode's data depending on its damping coefficient. However, we generally observe that higher frequency modes tend to be more complex than lower frequency ones, and this means more error could be tolerated in the representation. Thus, it may make sense to compress models using our methods, but use different error tolerances for each mode depending on its frequency (usually correlated with damping).

Low-Dimensional Bases for Modes of Cylindrically Symmetric Objects: Any cylindrically symmetric surface can be represented as a 2D profile-curve. This suggests that an even better compression scheme might be to represent the object's modes in terms of a low-dimensional basis consisting of a small set of displacement fields on the profile-curve. Specifically, each mode could be represented as an angular function $\mathbf{q}(\phi) \in \mathbb{R}^n$ that gives coordinates in the n-dimensional subspace. At run-time, we only need to store these functions, which can probably be repesented with a Fourier expansion, and the basis of profile-curve displacement fields.

Exploiting Reflective Symmetry: Along with cylindrical symmetry, we also examined objects with only reflective symmetry along a finite number of axes. Their modes also exhibit clear redundancy, including reflective symmetry and 2-fold rotational symmetry, so further compression may be possible by exploiting this.

Acoustic Transfer: Compressing the acoustic transfer model is also crucial, as it is very important for achieving high quality results [67], and models for fast evaluation can be very large as well. Transfer adds realism by computing how efficiently a given mode actually radiates in the surrounding space to a given listening point by solving the Helmhotz equation. This adds variety due to relative listening location, such as listening to a bell from below or from above, but it also reproduces important frequency characteristics caused by the shape of the object. Due to its constant-time evaluation cost, the FFAT map [23] is an appealing way of representing the transfer solutions, but such maps can be hundreds of megabytes in size. Zheng and James [144] use a single-point multiple expansion to approximate the transfer function, but it is unclear how much quality is lost due to truncation error.

Experimental Validation: Lastly, it is very likely that humans are actually unable to tell the difference between a low-error and high-error compressed sound model in practice. While a side-by-side comparison may exhibit obvious audible differences, there is the valid question of whether such differences affect the plausibility of the results. This is analogous to "visual equivalence" [114] in rendering, which quantifies equivalence not by numerical per-pixel comparisons, but rather by the perceived properties of the rendered materials. It would be fruitful to study how much error can be tolerated in a compressed sound model such that the synthesized sounds still remain "auditorily equivalent" to uncompressed models.

CHAPTER 5

MOTION-DRIVEN CONCATENATIVE SOUND SYNTHESIS OF CLOTH SOUNDS

In this final chapter, we present a practical data-driven method for automatically synthesizing plausible soundtracks for physics-based cloth animations running at graphics rates. Given a cloth animation, we analyze the deformations and use motion events to drive crumpling and friction sound models estimated from cloth measurements. We synthesize a low-quality sound signal, which is then used as a target signal for a concatenative sound synthesis (CSS) process. CSS selects a sequence of microsound units, very short segments, from a database of recorded cloth sounds, which best match the synthesized target sound in a low-dimensional feature-space after applying a hand-tuned warping function. The selected microsound units are concatenated together to produce the final cloth sound with minimal filtering. Unlike the methods presented in previous chapters, our approach here is data-driven. It avoids expensive physics-based synthesis of cloth sound, instead relying on cloth recordings and our motion-driven CSS approach for realism. We demonstrate its effectiveness on a variety of cloth animations involving various materials and character motions, including first-person virtual clothing with binaural sound. This work was originally published in An et al. [3].

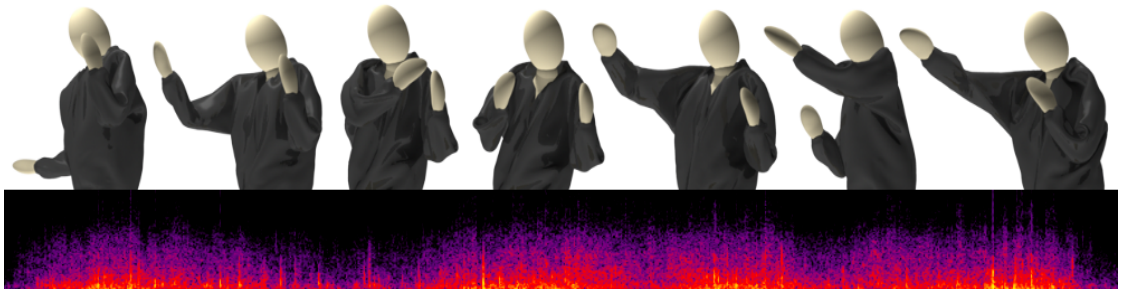


Figure 5.1: *Our data-driven approach to synthesizing cloth sounds is able to produce soundtracks for a wide range of common cloth animation scenarios. In this example, the familiar sounds of a windbreaker are synthesized as the character shadow boxes.*

5.1 Introduction

From the soft rustling of denim blue jeans or a woven cotton shirt, to the loud crumpling of a nylon windbreaker or the characteristic "zip" of corduroy pants, the natural sounds of clothing help bring virtual characters to life. Advances in computer graphics have enabled realistic visual simulation of cloth in computer animation and interactive virtual environments, but we still do not know how to automatically synthesize realistic sounds for synchronized accompaniment of these inherently silent cloth simulations.

Cloth animations pose unique challenges for digital sound synthesis. Cloth sounds are noisy, yet have a very natural and organic quality which is distinctively non-digital. People are also very familiar with clothing sounds, and so they can be quite attuned to the presence of digital synthesis artifacts. Direct numerical simulation of physics-based acoustic emissions from fabric is also highly complex, and would lead to expensive simulation times far beyond traditional cloth simulation. The need to simulate many different cloth materials, each with distinctive mechanical and sound properties, also complicates the estimation and tuning of parameters.

In this paper, we propose a *two-stage approach* for synthesizing cloth sounds that combines the responsiveness of motion-driven sound synthesis with the quality of real cloth recordings (see Figures 5.1 and 5.2). First, we devise a parametric cloth sound model driven by cloth motion that produces an initial *target sound* that captures important cues. Second, we use *concatenative sound synthesis* to piece together microsound *units*, from a database of pre-recorded cloth sounds, that best match the target sound.

Our parametric cloth sound model can be driven by input from graphics-rate cloth ani-

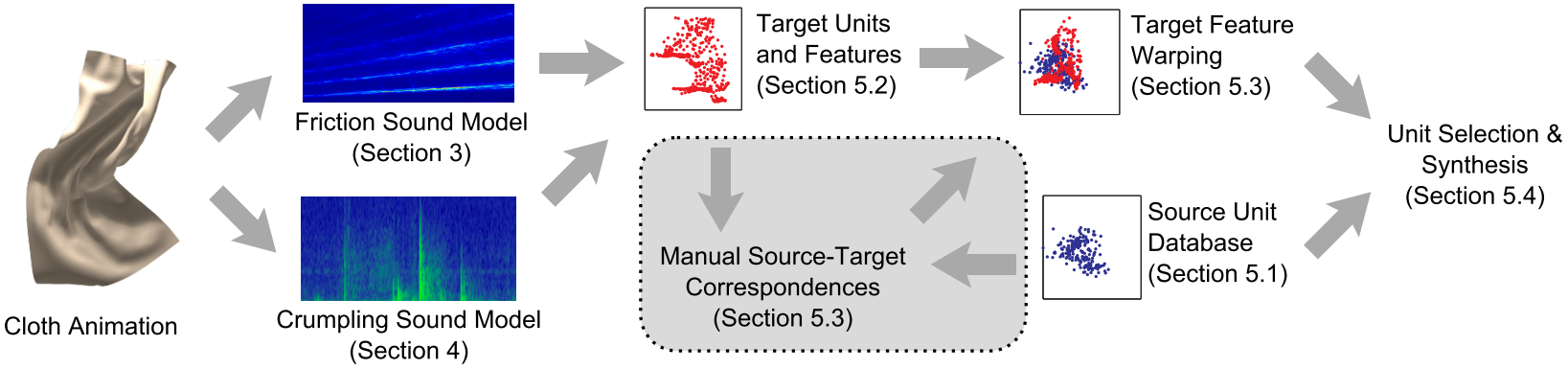


Figure 5.2: **Overview:** *Given a cloth animation, we first use two parametric sound models for friction and crumpling sounds to synthesize a low-quality "target" signal. We then dice the signal up into short sound "units" and compute per-unit feature vectors. Finally, we warp the features using a manually tuned warping function (which can be reused) and use a unit selection process to select a nearby high-quality "source unit" from a database to replace every target unit. The selected units are concatenated to synthesize the final cloth sound.*

mations. The model accounts for two primary sources of acoustic emissions from cloth: frictional contact sounds and crumpling sounds. The friction noise and crumpling sound models are built using data from experiments in which we record the sound of a specimen of a specific material as it undergoes specific motions. We then analyze the input cloth animation's motions, extracting sliding contact and curvature information, and use spectral and sample-based synthesis techniques to generate sound with characteristics matching those observed in the experimental data. The model produces a synchronized sound that mimics variations in the true sound, such as synchronized crumpling events, and friction noise dependence on sliding speed. However, due to model limitations, its realism is limited beyond specific controlled animations. Therefore, we only use this model to generate a low-quality *target signal*, which in turn drives the second synthesis process based on *concatenative sound synthesis* (CSS) [121]. CSS is used to piece together microsound *units* from a database of pre-recorded relevant sounds. We select a sequence of database units which best matches the sequence of target sound units. This *unit selection* process determines the distance between target and source units by using low-dimensional feature vectors based on mel-frequency cepstrum coefficients (MFCC). Since our synthesized target signal can differ significantly from the recorded database sounds, we warp the target's feature vector space with a manually tuned warping function to better match the variations in the database. The final synthesized cloth sound is then generated by concatenating the selected sound units together, with minimal filtering to avoid introducing digital artifacts.

Our data-driven method can produce plausible cloth sounds, and we can render realistic first-person cloth experiences by using low-noise binaural microphones for database recording. Our examples demonstrate its effectiveness with the challenging examples of corduroy pants and nylon windbreakers, as well as non-character examples such as

blankets and sheets (cotton and polyester). The CSS model is built using a specific material and garment undergoing particular actor motions along with a calibration dataset of simulated cloth motions; however, the model can be reused for novel simulated cloth motions at runtime.

5.2 Related Work

Cloth simulation is widespread in computer animation, and a variety of cloth models have been proposed and studied with the aim of visual reproduction of cloth behaviors [133, 37, 6, 73]. In addition to the underlying models, many methods and algorithms for integrating their dynamics also exist with varying trade-offs between efficiency and accuracy [6, 30], and strategies for resolving collisions and contact [18]. Unfortunately, such visual simulation methods are not inherently well-suited to resolving the acoustic vibrations of cloth.

Recently there has been increased interest in developing sound synthesis techniques for computer animations and virtual environments. Techniques vary in how much they are based on physical principles as opposed to recorded data. Many early synthesis techniques are based on simplified models of musical instruments, such as guitar strings and vibrating membranes [75, 14]. Recently, much work has been done on synchronizing physically based synthesis techniques with physically based animations. Rigid bodies are well-approximated by efficient linear modal synthesis techniques [138, 140, 103, 67, 112, 144], which unfortunately provide poor approximations to cloth sounds. Some types of clothing, for example plastic windbreakers, are well

modeled as thin shells. Nonlinear thin-shell and plate sounds have been widely considered (see [14, 23]), but such sound models are typically only valid for very small deformations and cannot support crumpling. O'Brien et al. [102] proposed a method for synthesizing sounds for general FEM-based simulations, and sheet-like examples were considered. However, due to high computational costs and difficult parameter tuning, we seek a method that is more practical and easier to control. It is also unclear if existing cloth models could model the mechanics of crumpling and friction between yarns well enough to produce plausible sound vibrations.

On the other end of the spectrum, data-driven methods focus on using, or reproducing characteristics of, recorded data when the underlying physical systems are either too expensive to simulate or methods simply cannot produce convincing sound [35, 107, 105]. These techniques often utilize general synthesis algorithms, such as inverse-FFT synthesis [119, 88], and additive and subtractive synthesis [124]. Our technique uses several of these techniques to synthesize low-quality target sounds based on sliding noise and crumpling events.

Sound texture modeling and synthesis methods can be used to resynthesize an audio corpus [43, 130]. Chadwick and James [24] synthesized fire sounds using a hybrid sound synthesis approach wherein a low-frequency sound is first generated from a physics-based simulation, then data-driven sound texture synthesis is used to add high-frequency details. In contrast, we use cloth motion to drive a low-quality measurement-based sound model, then use that to generate a target signal for use with concatenative sound synthesis. Our method shares some similarities with the "Sound-by-Numbers" algorithm, [22] which drives sound models using low-dimensional motion signals, such as the 2D position of a moving object. In contrast, 3D cloth animations produce high-

dimensional motion signals.

Inspired by granular synthesis techniques [118], our final sound is a concatenation of microsound units from cloth sound recordings. However, the particular selection of units is controlled by our low-quality synthesis model, and is most closely related to concatenative sound synthesis (CSS) [121]. CSS has origins in speech sound synthesis [65], where sound quality requirements dictates a data-driven approach. CSS has been used in computer music applications for musical instrument synthesis and for resynthesis of audio [121]. In such cases, the target signal can be symbolically represented, e.g., as a sequence of phonemes or musical notes, as well as using target audio, such as in audio resynthesis.

The particular topic of synthesizing cloth sounds has seen very little work. Cho et al. [28, 29] performed experimental studies concerning the characteristics of frictional sounds and how they affect the perceived qualities of fabrics. Huang et al. [64] synthesized sounds for a stylus being rubbed over a cloth patch using a modal model driven by measured roughness profiles. While appropriate for their particular haptic application, the model is unsuitable for general cloth animations. Crumpling sounds have received attention in the physics community due in large part to the interesting self-organized critical phenomena exhibited by acoustic emissions from crumpling events and their characteristic power-law statistics [63]. Such models have inspired geometry-independent stochastic sound models of crumpling [45].

In motion pictures and video games, cloth sounds are often generated using hand-selected recordings, or "acted out" by foley artists. Such approaches can produce high-quality results, but are not computer automated, e.g., for interactive virtual environ-

ments. When they are automated, such as when triggered by computer-generated events [132], the sounds lack nontrivial dependence on the cloth's deformation and contact state. In a sense, our approach automates the foley process by converting digital cloth motion directly into a plausible sequence of cloth recordings.

5.3 Friction Sound Model

Sliding friction is an important component of cloth sound and is common when cloth rubs against itself or other surfaces. In this section, we describe a frequency-domain noise model that provides a material-specific approximation of friction sound. A primary behavior we wish to capture is how the frictional sound changes with respect to sliding speed, since this can introduce pitch-like variations in many materials (see Figure 5.3). While other factors can be important, such as geometric shape, contact state, tension, and scattering, it can be difficult to devise practical experiments to model these variations. As a first approximation to friction-driven sound, we use experiments to build a colored-noise model with parametric dependence on sliding speed. The model suffices to synthesize a semi-plausible noise-like sound for cloth in sliding contact for purposes of generating a target sound.

5.3.1 Motion Analysis

Given the input cloth motion, our friction sound model is parameterized by sliding contact velocities. Sliding contact events can be generated by cloth simulators; however,

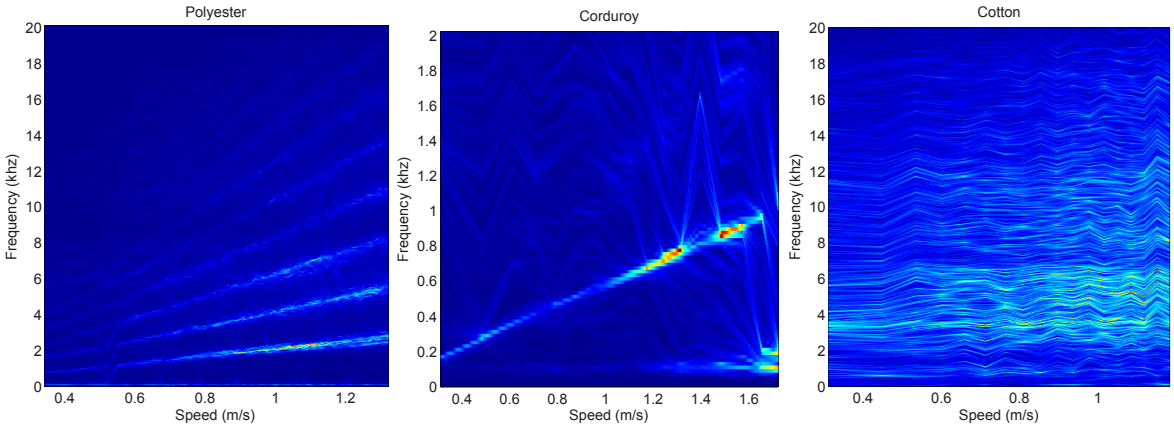


Figure 5.3: **Friction sound spectra vs. sliding speed** *are shown for several materials using the interpolated spectral sound model. Strong dependence on sliding speed is evident for some materials.*

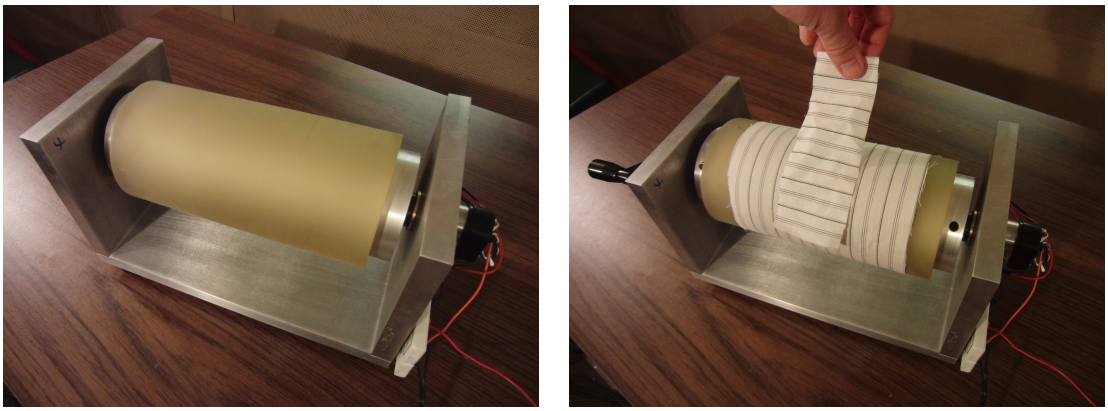
for generality, and for use with commodity simulators where contact state is often inaccessible, we choose to estimate sliding contact events via position-based collision analysis of animation frames. For simplicity, we only consider point-triangle contacts. To accommodate different contact gap tolerances and interpenetrations, we assume that each point has some finite collision radius r specified by the user. If there are triangles within distance r of a vertex, we record a sliding contact with the closest triangle, and estimate relative contact speed using a forward-difference scheme. To interpolate and avoid harsh discontinuities, we fit piecewise cubic Hermite interpolating splines to the raw sliding velocities. The result of the analysis is a function $s_v(t)$ that returns the speed at which vertex v was sliding at time t. If it was not in contact with anything, we set $s_v(t) = 0$ (which will produce no sound). Similarly we filter out all points which never slide above some given speed threshold s_{thresh}, setting their speed to zero, which helps eliminate contacts merely due to proximity in the rest pose. We observe that higher mesh resolutions tend to produce cleaner sound signals due to improved spatiotemporal sampling of contact events.

5.3.2 Experimental Analysis

To obtain calibrated measurements of cloth friction sounds versus sliding speed, we constructed a specialized apparatus for frictional cloth sound measurement (see Figure 5.4). We wrap a rectangular piece of cloth around a mechanical roller, then spin the roller while manually holding another smaller piece of cloth in contact with it. The roller slows as friction dissipates its momentum, and the resulting sound is recorded until the roller comes to rest. Because cloth sounds can be very quiet at low sliding speeds, it is important to minimize the presence of other sounds when recording an experiment. All our recordings were done in a sound isolation room (*Industrial Acoustics Company controlled acoustical environment*) using equipment with low levels of self-noise: *RODE NT1-A* microphones and the *TASCAM DR-680* digital recorder. High-quality bearings help minimize the additional noise of the device itself. An optical encoder attached to the roller is used to accurately estimate the rotational speed at any given time. Since the cloth wrapped around the roller has a seam which can introduce sound artifacts, we use the encoder information to ignore sounds during seam contact. This setup effectively gives us a mapping from sliding speed to friction sound.

5.3.3 Parametric Noise Model

We begin by extracting a number of short clips (50 ms in all our examples) from the recorded sound convolved with a triangular window and taking the amplitude Fourier transform of each. We look up the speed in the encoder data at the center of each clip's time interval, resulting in a set of $(s_i, \mathcal{A}_i)$ pairs, where s_i is the sliding speed and $\mathcal{A}_i(f)$



Device Usage

Figure 5.4: **Apparatus for measuring cloth friction sounds** *used to build a parametric friction noise model. A sample of cloth is wrapped around the roller, and another sample is held against it. The roller is spun and the friction sounds are recorded. An optical encoder on the roller provides synchronized sliding speed, $s(t)$.*

is the amplitude of the Fourier transform coefficient for frequency bin f. These single-speed spectra are then interpolated to construct a material-specific estimate of $\mathcal{A}(f,s)$ for continuously varying s (see Figure 5.3).

Given a sliding speed s, we interpolate the two nearest amplitude spectra, $\mathcal{A}_i$ and $\mathcal{A}_j$, with interpolation parameter $\alpha(s) = (s - s_i)/(s_j - s_i)$. Direct linear interpolation using $\mathcal{A}(f,s) = (1-\alpha)\mathcal{A}_i(f) + \alpha\mathcal{A}_j(f)$ produces an unsatisfactory effect of two different noise sources being blended together, which poorly captures pitch-changing "zipping" sounds. A better method is to interpret the spectra as probability distribution functions (PDFs), and linearly interpolate their *inverse cumulative distribution functions (CDFs)* instead [91] (see Figure 5.5). Specifically, we approximate $\mathcal{A}(f,s)$ using

$$c_k(f) = \int_0^f \mathcal{A}_k(g)\,dg, \tag{5.1}$$

$$c_\alpha^{-1} = (1-\alpha)\,c_i^{-1} + \alpha\,c_j^{-1}, \tag{5.2}$$

$$\mathcal{A}(f,s) = \frac{dc_\alpha}{df}(f). \tag{5.3}$$

This method provides a smooth and perceptually pleasing way to interpolate between

two amplitude spectra.

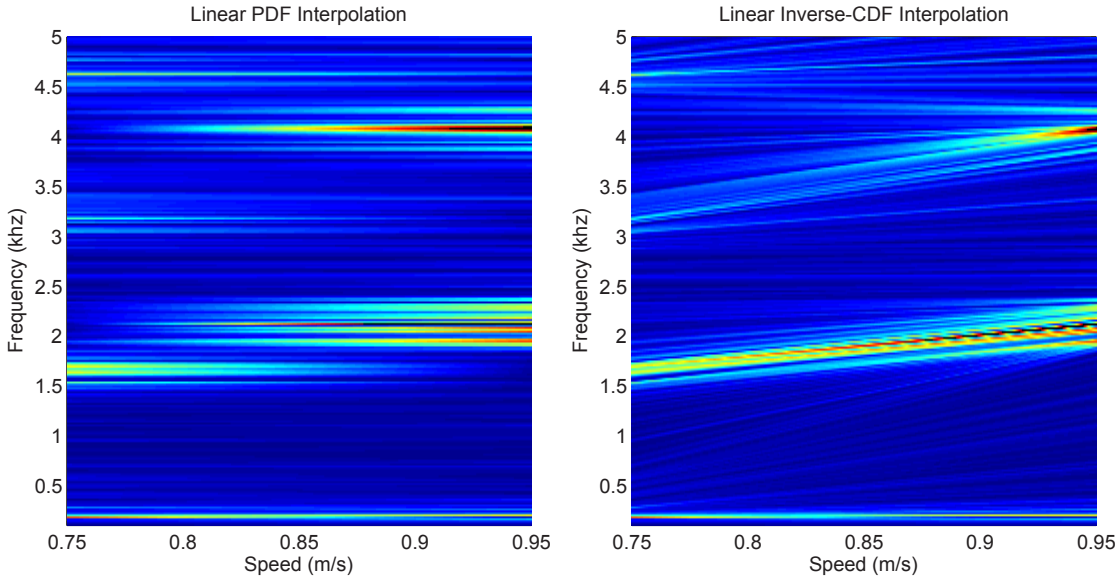


Figure 5.5: **Interpolating Noise Spectra:** *An example to illustrate the difference between interpolating the spectra as PDFs (Left) and interpolating the inverse CDF (Right). Interpolating PDFs corresponds to cross-fading two noise sources, whereas interpolating inverse CDFs yields a noise that shifts in pitch.*

5.3.4 Sound Synthesis

Using our parametric noise model, we can synthesize a soundtrack for each vertex v sliding with speed $s_v(t)$ at time t. We use the Inverse Fourier Transform (IFT) noise-synthesis method [124, 119, 88] where the output signal is synthesized as a sequence of 50ms clips that overlap by 25ms. For a given clip centered at time t, we assume a constant amplitude spectrum $\mathcal{A}(f, s_v(t))$. We synthesize noise using this spectrum by using random phases to yield complex frequency-domain coefficients, then perform the IFT. We also use the vertex's position at time t to perform any position-based auralization; in our examples, we apply an HRTF model [20]. Lastly, we overlap and add the clips. The final friction sound is the sum of all per-vertex friction sounds. While this

method ignores many other factors that add variation to frictional sound, such as transfer effects and variations due to tension and contact forces, it suffices to produce an initial low-quality target sound.

5.4 Crumpling Sound Model

As well as making frictional sounds, cloth can also buckle and produce crumpling sounds, which can sound like small "pops." Woven garments, such as dress shirts and denim jeans, produce audible crumpling sounds, and stiff synthetics, such as nylon windbreakers, exhibit characteristically loud crumpling sounds. By analyzing curvature changes in the input cloth animation, we estimate buckling events and an energy-like measure, and use this information to drive a data-driven crumpling sound model for target sound synthesis.

5.4.1 Motion Analysis

Given an input cloth animation, which simulates crumpling phenomena to varying degrees of accuracy, we analyze it to estimate the time, location and size of crumpling events. We resort to a simple heuristic based on mean curvature to decide when a crumpling event has occurred and how much energy was involved in it. We consider a vertex v to be "buckling" at frame time t if its mean curvature H_v^t changed sign from frame $t-1$ to t. If it changes from negative to non-negative, call it a "positive buckle," and the opposite direction is a "negative buckle" (see Figure 5.6). Once every positive (negative)

buckling vertex of frame t is identified, we take the graph consisting only of positive (negative) buckling vertices and edges incident to them. We find all connected components of such graphs and consider each component $\mathcal{C}$ to be a single event. Effectively, if a whole region of the cloth surface is buckling, this treats it as one large event rather than many small events. This produces a list of pop events $(t, \boldsymbol{p}, E)$, where $\boldsymbol{p}$ is the centroid of $\mathcal{C}$ and E is an energy-like measure of curvature changes

$$E = \left(\sum_{v \in \mathcal{C}} (H_v^t - H_v^{t-1}) \right)^2 = \| H_{\mathcal{C}}^t - H_{\mathcal{C}}^{t-1} \|_1^2. \tag{5.4}$$

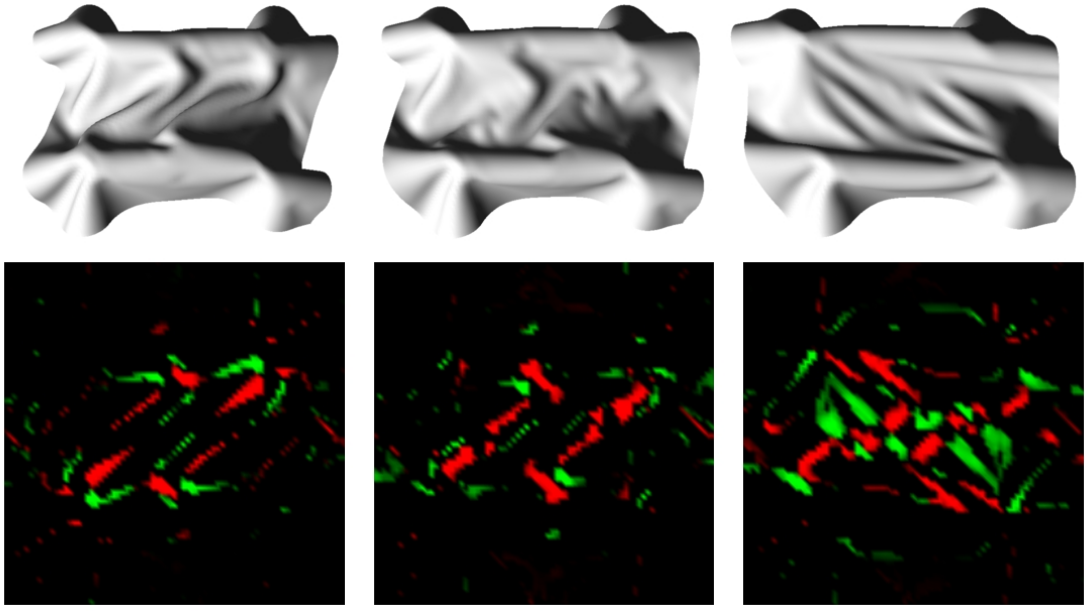


Figure 5.6: **Crumpling Motion Analysis:** *(Top) Three consecutive frames of a cloth simulation. (Bottom) Visualization of "buckling" vertices where black indicates no buckling, red a "negative buckle," and green a "positive buckle." Each contiguous region of red or green is treated as a single crumpling event.*

5.4.2 Experimental Analysis

We recorded crumpling sound events for use in data-driven sound synthesis. To isolate crumpling sounds, one must take care to minimize sliding contact sounds. We mount

a 30*cm* square of the cloth on two metal handles (using magnets to hold it in place), then manually deform the cloth using an up-and-down shearing motion (see Figure 5.7). This setup consistently produced many crumpling sounds without friction sounds. It also provides a direct path to the microphone, unlike the cylindrical method of [63]. We typically record about 10-20 seconds of crumpling.

Figure 5.7: **Crumpling Experiment:** *To record isolated crumpling sounds of a given material, we affix a square swatch to metal handles using strong magnets, then manually shear the sample.*

During post-processing we extract recorded samples of individual crumpling events using the same method as [63]: we convolve the energy (the sum of squared pressure values) with a 20ms rectangular window and look for consecutive runs with response greater than a threshold E_{thresh}. We then extend each sample's extents to zero-crossings of the signal. See Figure 5.8 for an example of extracted samples. The E_{thresh} values used for polyester, cotton, and the windbreaker were 0.25, 0.50, and 0.025, respectively. Buckling in corduroy was too quiet to merit modeling.

5.4.3 Data-driven Sound Synthesis

Given a crumpling event extracted from motion analysis, we assign a recorded crumpling sound with a similar "energy." We then simply play the sample when the event

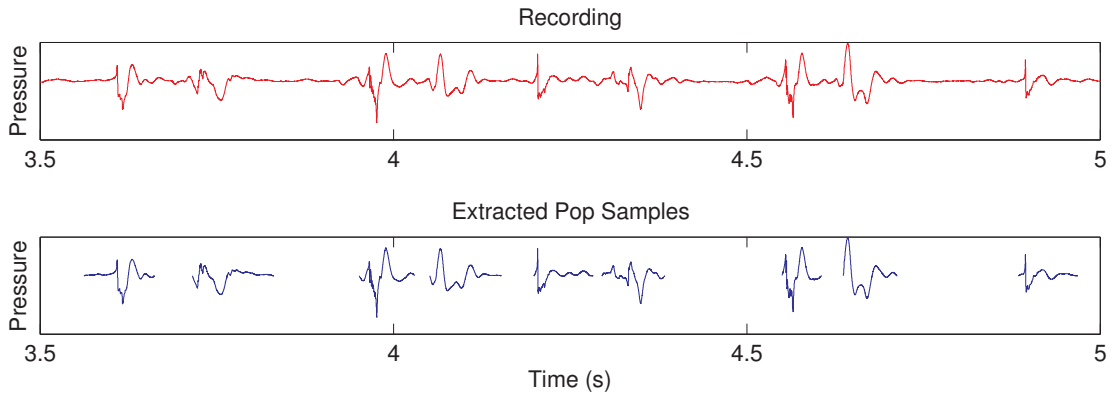


Figure 5.8: **Crumpling sound events** *extracted from a crumpling experiment recording (cotton) by an energy thresholding method.*

occurs to produce the output signal, with any subsequent auralization for stereo effects (we use an HRTF model [20]). To avoid frame-rate-dependent artifacts, we also randomly jitter the playback time by 1/30 second. Since the "energy" values from motion events and sound samples are very different quantities, we use histogram matching [57] to map an event energy E_e into a sample energy E_s, then use the sample with the nearest energy. Histogram matching essentially transforms the event distribution function $p_e(E_e)$ into the sample distribution $p_s(E_s)$ by evaluating $E_s = c_s^{-1}(c_e(E_e))$ where c_* is the cumulative distribution function (CDF) of p_*. We typically use 20 bins to compute the PDF, and approximate the CDF as a piecewise linear function. To reduce numerical noise artifacts and control the number of crumpling events, we reject all E_e events below a threshold E_{min}. Histogram matching can be done for each animation individually, but for online applications (where p_e is unknown *a priori*) we use p_e and E_{min} from a pre-existing "calibration animation" which contains representative crumpling motions.

5.5 Concatenative Synthesis of Cloth Sound

Given a cloth animation and the aforementioned friction and crumpling sound models, we can now synthesize a *target sound*,

$$X(t) = X_{friction}(t) + X_{crumpling}(t). \tag{5.5}$$

Unfortunately, while the target signal captures certain cloth characteristics and synchronized variations, it lacks the realism of real cloth recordings. Inspired by *Concatenative Sound Synthesis (CSS)* [65, 121], we construct an improved result by concatenating samples from a source database of relevant cloth recordings so that they best match the target sound. We dice both the target signal X and the source signal U into short sound *units*, $(x_i), i = 1 \ldots n$ and (u_j), respectively, using short non-overlapping windows (referred to as an arbitrary grain segmentation). For all materials except corduroy, we found 4.2ms units to be long enough to maintain the characteristics of the recorded sounds. For corduroy, we used 16.7ms units to accommodate its larger-scale temporal structure. We hypothesize that there exists a target-to-source unit mapping $j = J(i)$, found using *unit selection*, such that the signal of concatenated source units, $S = (u_{J(1)}\ u_{J(2)}\ \cdots\ u_{J(n)})$, is a plausible soundtrack for the cloth animation. To facilitate comparison of target and source units for unit selection, we compute descriptive feature vectors $\mathbf{f}(x)$ for each sound unit x (or u). Since the target and source signals can be quite different, we non-linearly warp the target feature vectors, $\mathcal{W}(\mathbf{f}(x))$, so that they better match the source database's feature vectors $\mathbf{f}(u)$ (§5.5.3). Finally, we perform unit selection to determine $J(i)$, essentially by minimizing the distance from $\mathcal{W}(\mathbf{f}(x_i))$ to $\mathbf{f}(u_{J(i)})$ (§5.5.4).

5.5.1 Database Acquisition

We construct databases using source recordings for specific materials, and particular cloth-character motion scenarios. For a richer database, we recorded each session using three microphones placed about a meter apart, and concatenated the signals into a single source signal. Images of the acquisition process are shown in Figure 5.9. Binaural recordings, which are ideal for self-sound listening experiences, were made using ultra-low-noise in-ear binaural microphones (*Sound Professionals, MS-TFB-2*). The exact motions recorded were chosen to cover calibration animations. For example, when recording the windbreaker, the subject wore a windbreaker and performed many jogging, flexing, punching, and waist-twisting motions. While a very large database can in principle improve the quality and range of the synthesized sound and avoid repetition, the target synthesis model is itself of limited fidelity so there are diminishing returns. In our examples, we typically recorded subject motions for about 1 minute.

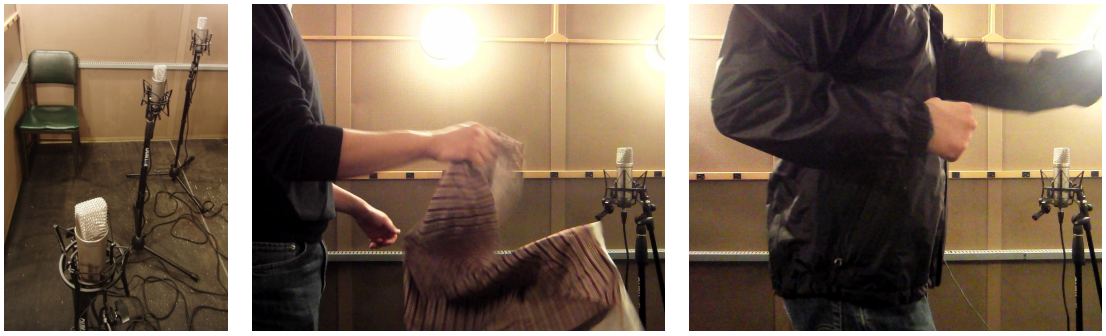


Figure 5.9: **Database acquisition:** *(Left) Various natural motions and garments were recorded in a sound isolation room. (Middle) A cotton sheet is lightly waved back and forth. (Right) Punching motions while wearing a windbreaker.*

5.5.2 Feature Vectors

Low-dimensional feature vectors are used to summarize target and source sound units and facilitate easy comparisons between otherwise high-dimensional sound units. We found that mel-frequency cepstrum coefficients (MFCC) (see [111]) provide effective descriptors for cloth sound units. The mel-frequency cepstrum is a popular compressed signal representation in speech applications. It works by taking the short-time Fourier transform, aggregating the spectral energy into uniform bins on the mel-frequency scale (an empirically derived scale meant to better match how humans distinguish pitch), and then taking a discrete cosine transform (DCT) of the logarithm of the bin energies:

$$\mathbf{f}(x) = \mathrm{DCT}(\log(\mathrm{mel}_N(x))) \tag{5.6}$$

where "mel" returns a spectral energy histogram with N bins spaced uniformly according to the mel scale.

This feature characterizes a unit's spectral shape and magnitude, and we find that $N = 3$ coefficients suffice to distinguish relevant characteristics of our sounds. Roughly speaking, the first coefficient correlates with loudness, and the other two coefficients correlate with how "crumply" a sound is (see Figure 5.10). It is unclear whether more coefficients would be helpful, and this is a topic of potential future work. For binaural recordings we concatenated the N-vector features of left and right channels to obtain a $2N$-vector feature.

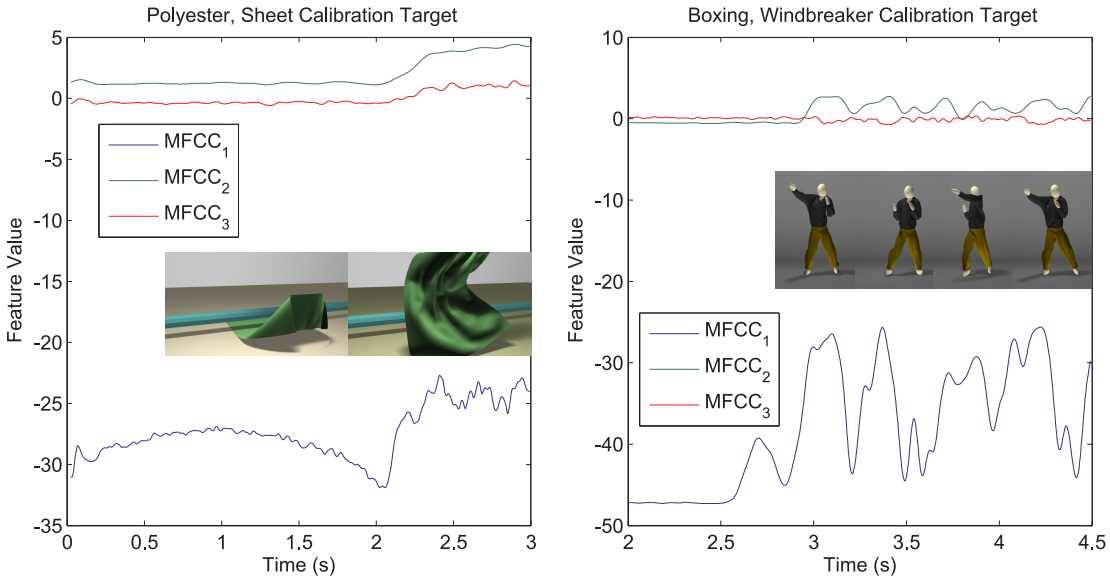


Figure 5.10: **MFCC features vs. time:** *Two plots illustrate how the 3 MFCC coefficients correlate with various motions. (Left) Unlike $MFCC_1$, $MFCC_2$ is unaffected by the variation in frictional sound for the first 2 seconds, but it fires when the shaking begins and crumpling sounds occur. (Right) We observe distinct peaks corresponding to punches and arm motion. Both $MFCC_1$ and $MFCC_2$ exhibit peaks for motions that are loud and crumply.*

5.5.3 Feature Warping

To facilitate unit selection, the feature-space distribution of target units must be warped to overlap well with that of the source units in the database. Given that these distributions can differ wildly depending on the target and recorded sounds, and that the "correct" target-to-database mapping is subjective and an opportunity to introduce stylization, we propose a user-guided approach. We synthesize a target signal for a set of training animations, then a sound designer provides a number of manual correspondences between parts of the target signal and the source/database signal. Given these correspondences, we fit a feature-warping function $\mathcal{W}(\mathbf{f}(x_i))$ based on thin-plate splines (see Figure 5.11) which can be reused for novel target signals. Filtering is used to make the process robust to noise and outliers. We now describe the process in detail.

Temporal Filtering: We temporally filter target and source features, since they can be noisy, e.g., due to crumpling events, and noisy features can lead to overfitting and bad feature matching for crumpling. We filter the features in time using a Gaussian filter; filter widths for each dimension are listed in Table 5.3.

Specifying Correspondences: Each correspondence is user-specified as time intervals (t_k, d_k, Δ_k), where for the k-th correspondence, t_k is the start of the interval in the target signal, d_k is the start in the database signal, and Δ_k is the length. For example, the most trivial correspondence is between intervals of silence in the target and database signals. Others might be between rapid sliding events and various degrees of desired crumpling. The intervals used in our examples are typically $\Delta = 100$ ms to $\Delta = 200$ ms long, and each fitting uses 5 to 15 correspondences. Figure 5.11 shows the interface we use for correspondence specification. Typically, it takes around 5 to 15 minutes of manual work to specify such correspondences, although it can take 2-5 iterations to achieve desirable results. See Section 5.7 for further discussion on this process. Given these correspondences (t_k, d_k, Δ_k), we find all units in these intervals and assume they correspond to each other in order. If $\{x_{i_1} \ldots x_{i_n}\}$ is the set of target units which lie within the time interval $[t_k, t_k + \Delta_k]$ and $\{u_{j_1} \ldots u_{j_n}\}$ is the set of database units in $[d_k, d_k + \Delta_k]$, then we assume the unit correspondences to be $C_k = \{(i_1, j_1) \ldots (i_n, j_n)\}$. We collect these lists of unit-pairs C_k for all given correspondences and concatenate them into C.

Warp Function Fitting: Given the n correspondences $C = \{(i_1, j_1) \ldots (i_n, j_n)\}$ between target and source units, we seek a smooth feature-warping function $\mathcal{W}$ that minimizes the error:

$$\sum_{k=1}^{n} \|\mathcal{W}(\mathbf{f}(x_{i_k})) - \mathbf{f}(u_{j_k})\|_2. \tag{5.7}$$

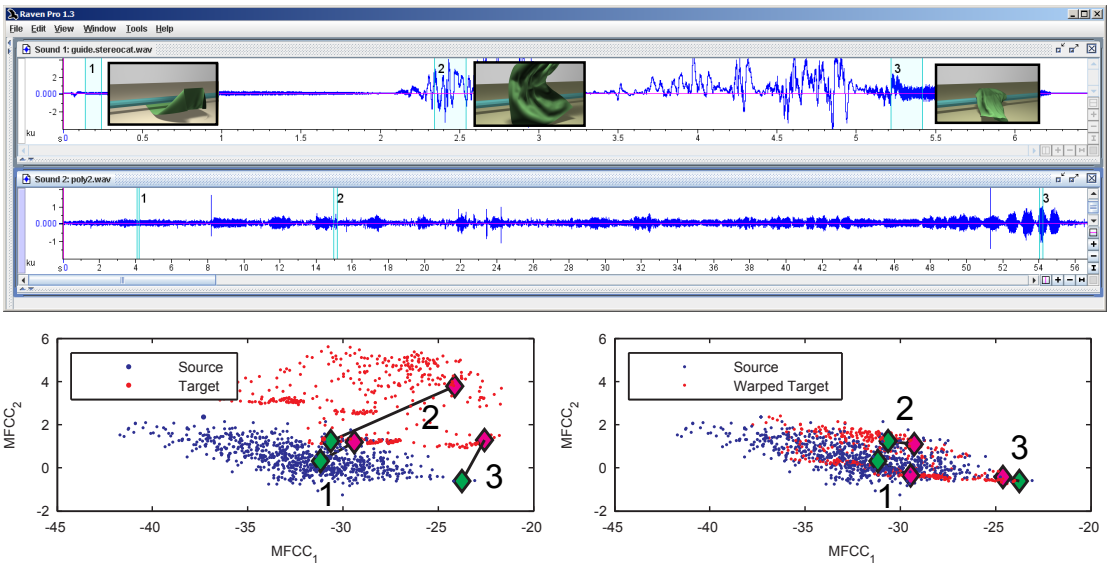


Figure 5.11: **Manual selection of sound correspondences** *is done using a simple interface (Top) with the target signal on top and the database source on the bottom. A sound designer selects target clips during perceptually important events, then selects database clips they would prefer to hear during such events. (Bottom) Plots of source and target units in feature-space (left/right showing before/after warping), along with larger markers at centroids of the selected clips. Target clips are magenta and source clips are green. Note that the correspondences are not exactly fit by the warping due to regularization.*

To warp the target unit features so that correspondences are respected, while also ensuring a smooth warp that avoids overfitting, we use a *regularized thin-plate spline* [141] similar to the approach taken by Belongie et al. [11]. For a d-dimensional feature $\mathbf{f} = [\mathrm{f}_1 \ldots \mathrm{f}_d]^T$, the displacement function for dimension l is

$$\Delta\mathbf{f}_l(\mathbf{f}) = A_{0,l} + \sum_{k=1}^{d} A_{k,l}\,\mathrm{f}_k + \sum_{k=1}^{n} W_{k,l}\,U(\|\mathbf{f} - \mathbf{f}(x_{i_k})\|_2) \tag{5.8}$$

where $U(r) = r^2 \log(r^2)$. The complete warping function is then

$$\mathcal{W}(\mathbf{f}) = \mathbf{f} + \Delta\mathbf{f}(\mathbf{f}). \tag{5.9}$$

The affine coefficients $A_{k,l}$, $k = 0 \ldots d$, and nodal weights $W_{k,l}$, $k = 1 \ldots n$, are obtained by solving a regularized linear system [11, 141]. We heavily regularize the system to avoid over-fitting and introducing unintended distortion; values for our regularization

parameter λ (from equation 10 of [11]) are given in Table 5.3.

Warp Reuse: In practice, we fit the feature warp once for a particular source database and calibration animation, then reuse it for novel target signals. This reuse reduces the need for manual intervention, and avoids example-specific tuning. Assuming the novel animations do not deviate too drastically from the feature-space covered by the calibration animation, the same warp will still produce plausible sounds. Thus artist intervention can be done just once per database–simulation pair to train the unit matching model, but an artist can still adjust warping to stylize the sound model.

5.5.4 Unit Selection & Synthesis

Next we select a database unit u_j for each target unit x_i by determining a selection function $j = J(i)$. Our final algorithm for unit selection is given below in Algorithm 2, which we now explain. Let the feature vector for the database unit u_j be $\mathbf{f}_j = \mathbf{f}(u_j)$, and let the warped and filtered target feature vector for unit x_i be $\tilde{\mathbf{f}}_i = \mathcal{W}(\mathbf{f}(x_i))$. The simplest unit selection method is to use the nearest neighbor given some distance metric:

$$J(i) = \underset{j}{\mathbf{arg\,min}}\, D_1(i,j) = \underset{j}{\mathbf{arg\,min}}\, \|\tilde{\mathbf{f}}_i - \mathbf{f}_j\|_2. \tag{5.10}$$

However, we find that it is often beneficial to use a contiguous sequence of units, $\{u_j, u_{j+1}, \ldots, u_{j+L}\}$ from the database to preserve important temporal structure of the original recording. For example, with corduroy pants, the zipping sound exhibits temporal structure larger than the unit size, but using larger units would restrict CSS flexibility. We use a simple greedy approach to encourage the selection of long sequences of units

by using the following distance metric:

$$D_2(i,j) = \sum_{k=0}^{L} D_1(i+k, j+k). \tag{5.11}$$

One can think of this as choosing a segment of the database unit curve, as traced out by the feature-space points $\{\mathbf{f}_j, \mathbf{f}_{j+1}, ..., \mathbf{f}_{j+L}\}$, that is closest to the warped target curve $\{\tilde{\mathbf{f}}_i, \tilde{\mathbf{f}}_{i+1}, ..., \tilde{\mathbf{f}}_{i+L}\}$). This tends to select more consecutive sequences of database units, and thus the resulting synthesized signal exhibits more of the original recording's temporal structure. In our examples, we use L values from 5 to 10 depending on material.

Avoiding large jumps: One issue that arises is that sometimes a database unit may be very far away in the D_1 metric, but close in the D_2 metric, resulting in undesirable "jump" artifacts which are difficult to blend. To avoid these cases, we limit our search to units u_j that are within a distance d_{max} of the warped target unit by using the modified distance function,

$$D_3(i,j) = \begin{cases} D_2(i,j) & \text{if } D_1(i,j) \leq d_{max} \\ \infty & \text{otherwise} \end{cases} \tag{5.12}$$

In all our examples, $d_{max} = 2.0$, except for the corduroy model where $d_{max} = 3.0$. Using D_3 ensures that the unit u_j chosen by the distance $D_2(i,j)$ will not be too far in terms of $D_1(i,j)$, and it also serves as an optimization, since we do not have to compute $D_2(i,j)$ for every database unit; a similar approach is taken by Pullen and Bregler [110]. On the off chance that no database units are within range, we fall back to the simple D_1 distance.

Avoiding repetition: Rapid repetition of selected units should be discouraged, since even repeating a unit once can produce a noticeable buzzing artifact. We explicitly avoid

this by giving every unit a "cool-off period" L_c: if a unit u_j is selected, it is not allowed to be selected for the next L_c target units. In all our examples, $L_c = 20$.

Algorithm 2: Match database units $j = J(i)$ to target units i

```
1  begin
       // Compute distance table T
2      for all pairs (i, j) do
3          T(i, j) ← D_3(i, j)
       // Fix up target units far from DB
4      Far ← {i | T(i, j) = ∞ ∀j}
5      for all pairs (i, j), i ∈ Far do
6          T(i, j) = D_1(i, j)
       // Compute matching J
7      for each target unit i do
8          J(i) = arg min_j T(i, j)
           // Enforce cool-off
9          for k ← 1 to L_c do
10             T(i + k, J(i)) = ∞
11     return J
12 end
```

Concatenating units: Simply concatenating selected source units can produce many inter-unit discontinuities. Large C^0 discontinuities in an audio signal are heard as undesirable "crackles." To ensure that each unit starts and ends at a zero crossing, we can extend each unit boundary to an adjacent zero-crossing, and blend the extra samples using an overlap-add to avoid making the signal longer than the original target signal. Zero-crossings can be sparse for signals with strong low-frequency content, so we apply a 5th-order Butterworth high-pass filter with a 100 Hz cutoff frequency. This filter does not noticeably affect the synthesis quality, and it makes for a simple solution to large C^0 discontinuities.

Table 5.1: **Example Timings:** *All timings were done on an 8 core 2.93GHz Intel Xeon processor and are reported in seconds.*

Cloth Type	Animation	Friction Analysis	Friction Synth.	Crumpling Analysis	Crumpling Synth.	Target Unit Features	Unit Select.
Cotton	Sheet	37	135	12	21	73	25
Cotton	Couch	104	310	16	13	132	48
Polyester	Sheet	37	295	12	66	74	33
Polyester	Couch	104	455	16	16	132	62
Windbreaker	Exercises	315	94	62	35	152	41
Windbreaker	Boxing	766	255	120	101	472	197
Windbreaker	Jogging	373	86	60	52	199	55
Corduroy	Exercises	55	90	-	-	45	5
Corduroy	Boxing	124	75	-	-	79	24
Corduroy	Jogging	58	80	-	-	37	6
Windbreaker	FP Boxing	766	274	120	94	446	65

5.6 Results

We now describe our results, but please watch and listen to the accompanying video (http://www.cs.cornell.edu/projects/sound/cloth/) with stereo headphones to hear the sonified animations. Figures 5.12 through 5.17 show rendered stills from all our example animations. Parameters related to specific animations and materials are given in Table 5.2, and timings in Table 5.1. Parameters and timings specific to CSS warps and unit selection are given in Table 5.3.

Implementation Details: The commercially available cloth simulator *SyFlex* was used along with *Maya 2009* for all our examples. They were simulated and rendered at 60 FPS, and the simulation times ranged from half an hour up to 4 hours for our longest and most challenging examples. Simulations were done with two Intel Xeon X5570 processors (2.9 GHz, 8 cores total) using 16 threads. Specific timings for other parts of the pipeline are given in Table 5.1, but the cloth simulation was typically the dominant job. Motion-capture data for character animations are from the CMU Motion Capture

Database. Geometry for the windbreaker jacket and pants are from *Poser 6*.

We record and synthesize all audio signals at 96 kHz. For all our examples except the binaural one, we essentially treat each stereo channel separately all the way through the pipeline. The results are different due to the use of a head-related transfer function, and they are combined in stereo in the video. The warps for the cotton and polyester sheets were trained using manual correspondences that covered both channels, as they were sufficiently different, but it sufficed to train only on the left channel for the character examples as the calibration animation is similar for both channels.

Lastly, our simulated examples often had contact speeds much higher than what we could measure in the friction sound experiment. In order to fully and evenly demonstrate the full range of friction sounds, we scaled down the speeds to keep them in range. For the character examples, due to the difficulty of simulating the stiff and light windbreaker material, we further applied a non-linear function to the sliding speeds in order to subdue lower–speed contacts that were causing excessive noise: $\bar{s}_v(t) = (s_v(t)/4)^{1.5}$. For speeds that were below our measured models, we quadratically scale down the lowest-speed spectrum $\mathcal{A}_0$ towards zero: $\mathcal{A}(f,s) = (s/s_0)^2 \mathcal{A}_0(f)$ for $s < s_0$.

EXAMPLE (Cotton Sheet): This calibration animation exercises a cotton sheet by dragging it on the floor, picking it up, dropping it, and dragging it over a cylinder. Manual correspondences were specified at various points of crumpling and sliding, and particular care was taken to make sure the sliding motions produced no crumpling. The floor and cylinder were treated as being covered in cotton as well, as we do not handle friction sounds between multiple material types.

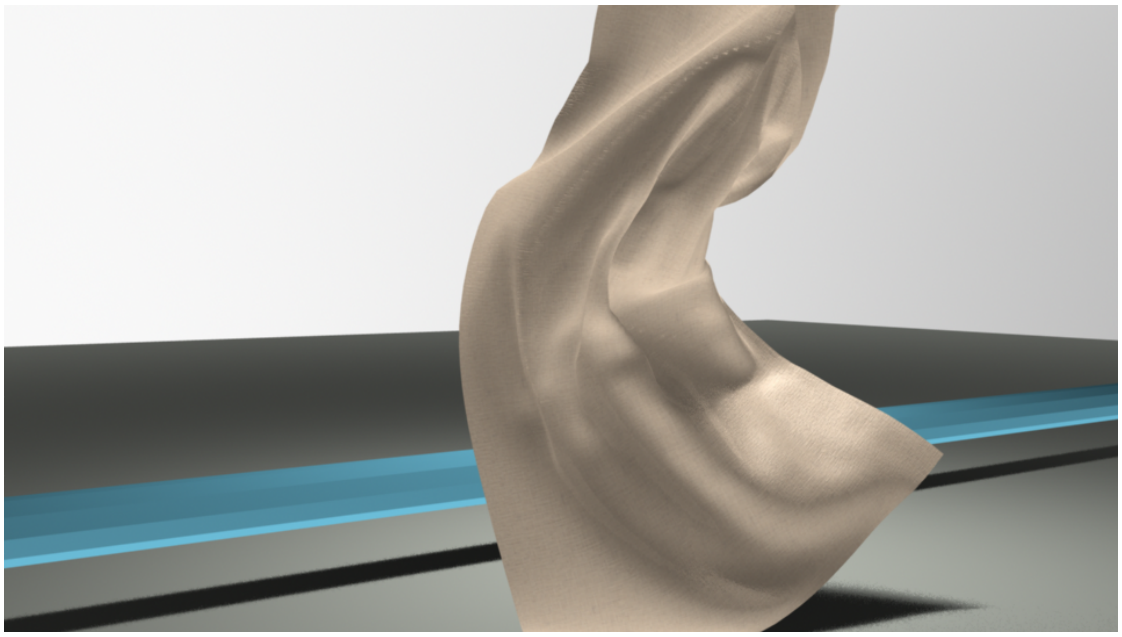


Figure 5.12: **Cotton Sheet:** *An animation of a cotton sheet being dragged over a floor and a bar.*

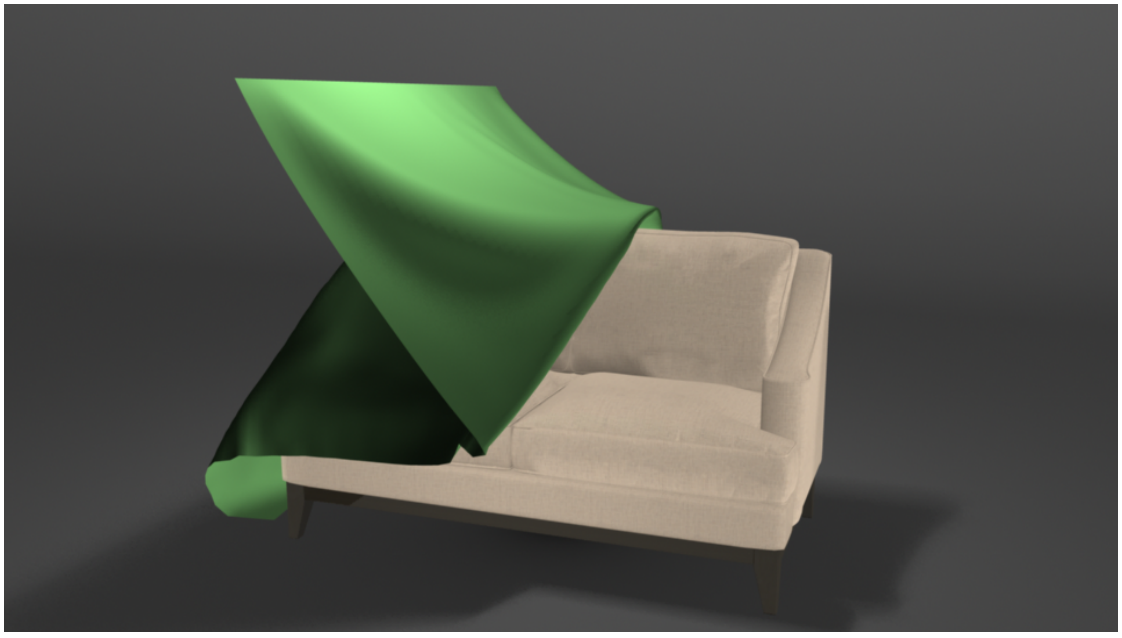


Figure 5.13: **Polyester Sheet & Couch:** *An animation of a polyester sheet being draped and dragged over a couch.*

EXAMPLE (Cotton Couch): A sheet is draped over a cotton-lined couch, peeled off, and then dragged over again rapidly. It uses the warping from the Cotton Sheet example,

so no manual intervention was required for CSS. Notice the characteristic crumpling at the end when the sheet quickly slips off the couch.

EXAMPLE (Polyester Sheet & Couch): We used the same animations as the previous two examples, except we synthesize them as polyester. The resulting sound is noticeably different, with higher pitched narrow-band content, as is characteristic of light polyester. When the sheet quickly slips off the couch at the end of the couch example, the sliding noise quickly shifts up in pitch, giving a distinct "zip" effect.

Figure 5.14: **Exercises:** *A motion-captured animation of a person performing exercises.*

EXAMPLE (Exercises): A motion-captured character goes through a variety of exercises and stretches while wearing a nylon windbreaker and corduroy pants. This serves as the calibration animation for these materials, so we manually tuned a warping and reused it for the next two examples (Jogging and Boxing). Some of the motion was edited in order to cover enough of the feature-space, such as the second half of the in-

place jogging being sped up, so some parts may look artificial. It should be noted that we do not use the crumpling histogram match from this for the next two examples, as the crumpling energy distributions were too different.

Figure 5.15: **Jogging:** *A motion-captured animation of a person jogging in a circle.*

EXAMPLE (Jogging): A character jogs back and forth a few times to demonstrate spatialization effects (HRTF [20] and distance falloff). CSS automatically selects quieter source units for quieter target units, so the spatialization is preserved. Some jog cycles do not produce the expected "zip" of the corduroy pants due to the fact that the legs are often separated in the motion capture data.

EXAMPLE (Boxing): A character "shadow boxes," throwing a variety of punches. This demonstrates the strong synchronization between the synthesized sound and the animation. The corduroy pants are relatively quiet due to the boxer's wide stance.

Figure 5.16: **Boxing:** *A motion-captured animation of a person shadow-boxing.*

Figure 5.17: **First-person Boxing:** *A motion-captured animation of a person shadow-boxing, rendered from a first-person perspective. The source database was recorded using binaural microphones to capture the sound of the windbreaker from the wearer's ears.*

Table 5.2: **Example Parameters:** *The parameters used for the windbreaker in "boxing" were also used for "first-person boxing."*

Cloth Type	Animation	Collision radius r	s_{thresh}	E_{min}
Cotton	Sheet	0.05	0.02	4878
Cotton	Couch	0.2	0.02	4878
Polyester	Sheet	0.05	0.02	528
Polyester	Couch	0.2	0.02	528
Windbreaker	Exercises	0.05	0.0	98902
Windbreaker	Boxing	0.05	1.0	172875
Windbreaker	Jogging	0.05	0.0	906402
Corduroy	Exercises	0.05	2.0	-
Corduroy	Boxing	0.05	1.0	-
Corduroy	Jogging	0.05	2.0	-

EXAMPLE (First-person Boxing): The same boxing animation as in the previous example, but we position the camera and the listening position on the head of the character. In order to capture the sound of actually wearing a windbreaker—which is very different from hearing a windbreaker from a distance—we recorded a different database using the binaural microphones. Notice subtle spatialization effects in the final CSS result. The corduroy pants are not included in this sound.

Table 5.3: **CSS Model Parameters and Timings:** *Each calibrated CSS model consists of a TPS warp fit to manual correspondences along with unit selection parameters. The animation listed is the calibration animation, and the same parameters were used for other animations that reuse the CSS model. n is the number of manual correspondences, and the number of correspondence units is effectively the number of TPS nodes used for warping. The time to perform TPS fitting was negligible. All manual correspondences were clips of length 0.1 or 0.2 seconds long. Gaussian filter widths are given in number of sound units, and the standard deviation was a quarter of the width. The last column is how long it took to compute features for the database, in seconds.*

Cloth Type	Animation	n	# Corr. units	TPS λ	L	Gaussian widths	DB Length (m:ss)	DB Features
Cotton	Sheet	14	419	1.0e2	5	20, 50, 10	3:45	1317s
Polyester	Sheet	9	280	1.0e5	10	10, 50, 20	4:05	1392s
Windbreaker	Exercises	5	187	1.0e5	5	20, 30, 10	3:13	1101s
Corduroy	Exercises	3	15	0.0	10	20, 30, 10	2:41	235s
Windbreaker	FP Boxing	8	208	1.0e5	5	20, 30, 10	1:32	1051s

5.7 Conclusion

We have presented a data-driven method for automatic concatenative synthesis of sounds for 3D cloth animations. We focus on two specific sound-producing phenomena, friction and crumpling, and we have demonstrated that these are sufficient for a variety of animated cloth scenarios. Our data-driven method is computationally efficient and requires a reasonable amount of recorded data and human intervention. This makes it practical for pre-rendered movies, and it has potential to be fast enough for interactive virtual environments in the future. Although it is difficult to make precise, objective measurements of accuracy, our results do compare favorably to actual recordings.

Limitations & Future Work: Our focus on friction and crumpling sounds is motivated by the observation that these two sound sources dominate acoustic emissions arising from character clothing motions, but for high-speed cloth animations, such as a flag waving in the wind or the whipping of bed sheet, other emissions due to increased tension and impacts become important. One potential approach is to do low-resolution simulations of such motions to get these tension-based sounds, where the cloth momentarily acts like a vibrating membrane, and add this to the target signal. Our friction sound model also does not account for other sources of variation, such as contact pressure and sliding direction.

Our current pipeline does not allow for interactions between multiple types of cloth. Our features are not adequate for distinguishing between, for example, cotton contacting cotton versus cotton contacting polyester. More sophisticated features may be able to distinguish the two, allowing the target signal to contain a mix of various cloth types.

The interface for manual feature correspondences could be improved to be more intuitive. Currently, depending on the variety of sounds involved in a given example, finding a good set of manual correspondences can take many iterations of trial and error, sometimes up to 5. We conservatively estimate that each warping function in our examples took about 1-3 hours to tune, including manual correspondences and synthesis of the calibration animation to assess the warping with various TPS λ values. Specifying the correspondences only takes about 5 to 15 minutes per iteration (please see the supplemental video for a typical session), and the rest of the time is for the CSS pipeline. The time to specify correspondences and total number of iterations may vary depending on the experience level of the user. Future work will focus on providing the user with rapid and intuitive feedback as to which correspondences are causing undesirable synthesis results, and the system should also suggest correspondences based on analysis of the feature distributions.

Other potential areas for future research include acoustic transfer, such as occlusion of sound emissions by large bodies. It is unclear how to approach the problem of acoustic transfer around highly deformable cloth, and it is also unclear how much it matters. Also, we fully simulate clothing in our character examples, but it may be desirable to produce sound for skinned cloth models that lack proper sliding and crumpling events. Lastly, we use a local greedy unit selection algorithm for its potential to be used in online applications, but it is likely that global optimization techniques would provide higher quality results for prerendered movies.

CHAPTER 6

CONCLUSION

This thesis has presented three contributions which aim to advance the state of the art in sound synthesis techniques for virtual environments. A method for efficient simulation of reduced-order dynamics was presented, and it was applied to the difficult problem of synthesizing thin shell sounds. Thin shells are known to be difficult and expensive to simulate at audio-rates with prior methods, but the use of optimized cubature enabled dramatic gains in efficiency. The method successfully synthesized plausible and interesting sounds for objects such as trash cans, crash cymbals, and plastic containers. To address the memory costs of modal sound models in general, some compression techniques were proposed to make them more practical for interactive applications. It is hoped that this will result in more widespread adoption of such models in scenarios where memory is scarce, such as video games and training simulations. Lastly, a data-driven method was applied to the case of cloth sounds, a particularly difficult problem because human ears are so familiar with them. While a more general physics-based method was not achieved due to limitations of existing cloth models, the method presented is practical and can be used with any cloth simulator. Convincing soundtracks were synthesized for animations of large sheets being dragged over furniture and character animations featuring corduroy pants and nylon windbreakers. These contributions expand the range of real-world phenomenon that can be realistically sonified, particularly in the class of highly deformable bodies.

As a relatively new field, many problems still remain to be tackled. In general, many physics-based methods are still too expensive for practical use. The thin-shells method presented in this thesis, while it is a major improvement over existing methods, is cer-

tainly not suitable for real-time applications. Many other physics-based methods share this common issue. While hardware performance will certainly improve, possibly by surprising amounts, the community cannot rely on it alone to solve such performance problems. There is plenty of room in the field for more efficient algorithms, approximations with better trade-offs, and completely new pipelines. For data-driven methods, peformance is generally better, but they typically require more manual intervention and more experience with the system in order to reliably produce good results. This is an area that could use better user interfaces, more intuitive pipelines, and faster algorithms for more rapid iteration. Even for physics-based methods, intuitive user control and stylization of synthesis results remains an open problem. It is well-known from the visual rendering community that realism is not always the end-goal for users, and it is important for the sound rendering community to address such issues.

It would be beneficial to have such sound synthesis techniques be adopted for the interactive applications of the future. Currently, most sounds in video games and virtual reality simulations are hand-tuned and triggered with ad-hoc methods, and this often reduces the believabilty of the environments when the same sounds are heard repeatedly. By using synchronized synthesis methods where appropriate, such simulations can achieve a higher level of realism and immersion, resulting in more engaging and useful experiences. Adoption has occurred in some cases [84], but it is not widespread. While the benefit to pre-rendered movies is less obvious, there is still a case to be made for the cost-savings, as foley can be an expensive and time-consuming process. Ultimately, it is unlikely that synthesis algorithms will completely replace hand-tuned, recorded sounds and skilled sound design, and this is certainly not the intent of this research. But these are tools that will hopefully complement existing techniques, enabling creative individuals to fullfil their visions in new, innovative ways.

Sound synthesis is an exciting field with plenty of open problems and high potential for future impact. While analogies can be made with visual appearance modeling, and in many cases techniques can be shared between the two, the fundamental difference between sound and light make sound synthesis an interesting field with its own unique challenges. This thesis has presented a few advances to the field, and hopefully future work will continue to push the boundaries of what is possible.

APPENDIX A

PERCEPTUAL WEIGHTS FOR MODE SIMPLIFICATION

One improvement we explored for mesh simplification (Section 4.4) was to guide the simplification not only by modal values, but also by their importance. The hope was that some modes, for various reasons, may not actually be as important as other modes. We implemented this, but unfortunately results were not noticeably better. We include this exposition as a part of this thesis so others may see what we tried and possibly improve upon it.

A.1 Perceptual Weights Derivation

To gain further compression, we wish to derive weights that indicate how important one mode is relative to others. These weights can then be used to guide the simplification algorithm to generate a mesh that optimizes more for the important modes and less for the unimportant ones. To judge the importance of a mode we take into account three main observations:

1. For most rigid body simulation scenarios, impact forces tend to be pointed almost along the surface normal, and tend to last for a very short duraction. Thus, they can be well-approximated by an instantaneous impulse along the surface normal direction with some scale. This gives us an idea of how the modal system tends to be driven for a given surface vertex.
2. Some modes will not radiate efficiently at all due to object geometry, and thus their acoustic transfer function will have values far lower than other modes. These

modes should be weighed less.

3. Masking is a well-studied psychoacoustic phenonmenon where a tone of one frequency will mask out softer tones of nearby frequencies [146]. If one mode tends to be masked out by other modes, it should be weighed less.

We combine all these observations in the following manner: For every vertex v, we want relative weights to represent each mode's importance relative to others. We start by considering the representative case of a force in the direction of the surface normal $\hat{n}_v$, so $\mathbf{f}_v = s\hat{n}_v$, where s is a scalar. This means the modal force $\tilde{\mathbf{f}}$ resulting from $\mathbf{f}$ takes the form $\tilde{\mathbf{f}} = s\mathbf{U}_v^T\hat{n}_v$. This allows us to reason about the *relative* modal force each mode is likely to experience as a result of an impact on v, since $\mathbf{U}_v$ and $\hat{n}_v$ are constant. The relative force between two modes i and j is then:

$$\frac{\tilde{\mathbf{f}}_i}{\tilde{\mathbf{f}}_j} = \frac{\mathbf{U}_{v,i}^T\hat{n}_v}{\mathbf{U}_{v,j}^T\hat{n}_v} \tag{A.1}$$

Using this, we can derive relative pressure ratios $\frac{\tilde{p}_i}{\tilde{p}_j}$ as well under certain representative conditions and exploting the transfer function (the full derivation is given at the end of this section). Then for each mode i, we compute the probability that it will be masked by other modes after a normal impact on vertex v. This probability is what we use as the weight $w_{v,i}$. To reason about this, we consider a reasonable sound pressure level (SPL) range of 1 dB to 70 dB that mode i might be driven to. For what fraction of that SPL range is mode i audible, meaning it is not masked by the other modes? This can be done by assuming mode i is being driven to some SPL L_i and then using our relative pressure ratios to obtain L_j for all other modes j. Our masking model then tells us if mode i would be masked by any other mode. We refer the reader to [139] for the rationale behind the masking model that we chose to adopt. The parameters we use are $a_v = 20$ dB, and we also lower the threshold of hearing in quiet by 3.5 dB to be conservative.

We breakdown the weights even further by $c \in x, y, z$ components to account for cases where certain modes will probably not even be forced by a force in the normal direction because the mode's displacement is in the tangent plane. So the final weight used for $\mathbf{U}_{v_c,i}$ is actually $\lambda_{v,c,i} = |\hat{n}_{v,c}| w_{v,i}$. Lastly, we normalize all weights for a vertex v so the highest weight value is 1.0. Figure A.1 plots typical weight distributions for the heptoroid example.

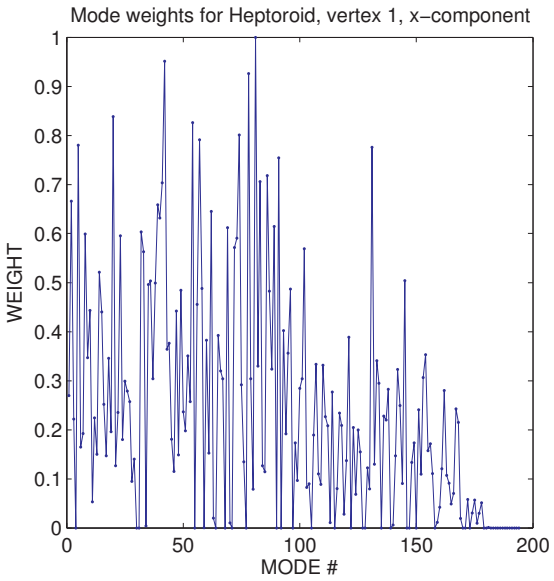


Figure A.1: **Typical Weights:** *Modal weight distributions are plotted for all modes of the bronze heptoroid example. The average, minimum, maximum, and standard deviation bars are taken over all vertices and all components for each mode. Note that many modes are weighted to 0, meaning they are always masked by other modes. This gives the simplification algorithm more freedom to accomodate the modes that are less masked, thus resulting in smarter optimization and selection of edge collapses.*

Relative Pressure: Suppose that for two modes i and j, we have relative a force ratio $\frac{f_i}{f_j}$. We now show how to derive the relative root mean square pressure ratio $\frac{\tilde{p}_i}{\tilde{p}_j}$ at a given listening location. To do so, we will use the sound power P of a given mode to derive a principled estimate of its typical $\tilde{p}$ over space given the root mean square mode state $\tilde{q}$. If the angular variation is small and the sound source is well approximated by a

spherically spreading source past an adequate distance, the pressure can be considered the same in all directions when integrating over a sphere $\mathbb{S}$ of radius r. Thus, when integrating intensity over the sphere to compute power, $\tilde{p}$ is constant:

$$P = \int_{\mathbb{S}} < I_n >_t \, d\mathbb{S} = \frac{1}{\rho c} \int_{\mathbb{S}} \tilde{p}^2(r) d\mathbb{S} = \frac{\tilde{p}^2(r)}{\rho c} 4\pi r^2 \tag{A.2}$$

Given that $\tilde{p}(r) = T(r)\tilde{q}$, where $T(r)$ is the acoustic transfer value dependent only on listening position (which, assuming angular invariance, is the radius r), we have:

$$P = T(r)^2 \tilde{q}^2 \frac{4\pi r^2}{\rho c} \tag{A.3}$$

To solve for $T(r)$, which is independent of $\tilde{q}$, we use the FastBEM solution that gives us sound power $\hat{P}$ for a unit displacement of $q = 1$, or $\tilde{q} = 1/\sqrt{2}$:

$$T(r)^2 = \frac{\rho c \hat{P}}{2\pi r^2} \tag{A.4}$$

Combining the last two equations gives us $P(\tilde{q}) = 2\hat{P}\tilde{q}^2$, and combining this with Equation (A.2) gives:

$$\tilde{p}(r, \tilde{q}) = \frac{\tilde{q}}{r} \sqrt{\frac{\hat{P}\rho c}{2\pi}} \tag{A.5}$$

For the common case where impact forces are well-approximated by impulses, the immediate RMS state $\tilde{q}$ of the impulse response is given by:

$$\tilde{q} = \frac{fh}{m\omega\sqrt{2}} \tag{A.6}$$

Where m is the mode's mass, ω the mode's angular frequency, and h the simulation timestep. Given the last two equations, it is trivial to derive the relative pressure ratio $\frac{\tilde{p}_i}{\tilde{p}_j}$ given relative force ratio $\frac{f_i}{f_j}$. The radius r and timestep h cancel out, and all other constants are known.

A.2 Simplification with Per-vertex Weights

We extend the surface simplification algorithm presented in [61] by adding per-vertex per-attribute weights. The original algorithm handles per-attribute weights that are uniform at every location. However, in our case, different attributes (modal values) may be important depending on the location of the model. Equation 4.5 then becomes:

$$Q^f(v = (p, s, \lambda_v)) = Q_p^f(v) + \sum_{j=1}^{m} \lambda_{v,j}^2 Q_{s_j}^f(v) \tag{A.7}$$

When we perform a collapse between vertices v_1 and v_2 into v, we must assign new weights to v. We currently just take the average of the two, so:

$$\lambda_v = \frac{1}{2}(\lambda_{v_1} + \lambda_{v_2}) \tag{A.8}$$

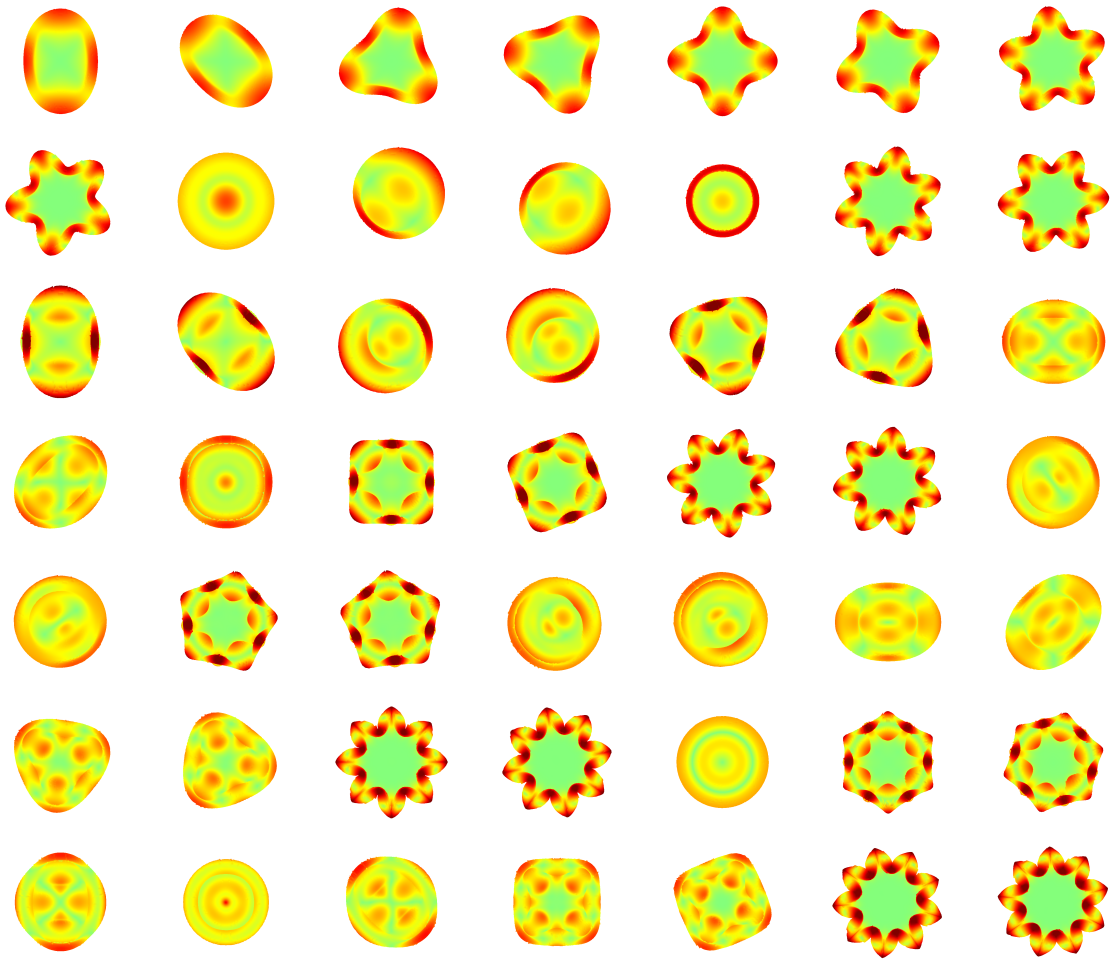
We also re-normalize after doing this average. Because we recompute the error quadrics for every collapse using the averaged weights, we must use the so-called "memoryless" version of quadric error simplification. Fortunately, Hoppe [61] reported that this was actually favorable and gave better approximations without much additional cost.

Geometry vs. Attribute Weighting: One parameter in quadric error simplification is the overall weighting between geometry and attributes. Typically, the user specifies a single scalar weight Λ indicating the importance of interpolating attributes versus approximating the geometry. To ensure size invariance for this value between different geometries, the geometry is scaled to fit a unit cube before simplification. Via some experimentation, we found that using a weight of $\Lambda = 1.0$ gave the most consistently best results amongst various examples.

APPENDIX B

MODE VISUALIZATIONS FOR PLASTIC BOWL

Here we present all 49 modes of the plastic bowl model from Section 4.6.2 sorted by frequency in row-major order. These are rendered from a top-down perspective, the geometry is displaced according to the mode's displacement field, and the color corresponds to the magnitude of the displacement.



BIBLIOGRAPHY

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] Jean-Marie Adrien. The missing link: Modal synthesis. In *Representations of musical signals*, pages 269–298. MIT Press, Cambridge, MA, USA, 1991.

[3] Steven S. An, Doug L. James, and Steve Marschner. Motion-driven concatenative synthesis of cloth sounds. *ACM Transactions on Graphics (SIGGRAPH 2012)*, August 2012.

[4] Steven S. An, Theodore Kim, and Doug L. James. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.*, 27(5):1–10, 2008.

[5] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 303–308, July 1992.

[6] David Baraff and Andrew P. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54, July 1998.

[7] Jernej Barbič. *Real-time Reduced Large-Deformation Models and Distributed Contact for Computer Graphics and Haptics*. PhD thesis, Carnegie Mellon University, 2007.

[8] Jernej Barbič and Doug L. James. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. on Graphics*, 24(3):982–990, August 2005.

[9] Jernej Barbič and Doug L. James. Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation (SCA 2007), San Diego, CA*, August 2007.

[10] Klaus-Jürgen Bathe. *Finite Element Procedures*. Prentice Hall, second edition, 1996.

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 24(4):509 –522, April 2002.

[12] M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, and E. Grinspun. A quadratic bending model for inextensible surfaces. In *Eurographics Symposium on Geometry Processing*, pages 227–230, 2006.

[13] S. Bilbao. A family of conservative finite difference schemes for the dynamical von Karman plate equations. *Numerical Methods for Partial Differential Equations*, 24(1), 2008.

[14] S. Bilbao. *Numerical Sound Synthesis*. Wiley Online Library, 2009.

[15] Javier Bonet and Richard D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, New York, second edition, 2008.

[16] Nicolas Bonneel, George Drettakis, Nicolas Tsingos, Isabelle Viaud-Delmon, and Doug James. Fast Modal Sounds with Scalable Frequency-Domain Synthesis. *ACM Transactions on Graphics*, 27(3):24:1–24:9, August 2008.

[17] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 28–36, August 2003.

[18] Robert Bridson, Ronald P. Fedkiw, and John Anderson. Robust treatment of collisions, contact, and friction for cloth animation. *ACM Transactions on Graphics*, 21(3):594–603, July 2002.

[19] C. Phillip Brown and Richard O. Duda. A Structural Model for Binaural Sound Synthesis. *IEEE Trans. on Speech and Audio Processing*, 6(5), 1998.

[20] C.P. Brown and R.O. Duda. A structural model for binaural sound synthesis. *Speech and Audio Processing, IEEE Transactions on*, 6(5):476 –488, sep 1998.

[21] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. A Multiresolution Framework for Dynamic Deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 41–48, July 2002.

[22] M. Cardle, S. Brooks, Z. Bar-Joseph, and P. Robinson. Sound-by-numbers: motion-driven sound synthesis. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 349–356, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[23] Jeffrey N. Chadwick, Steven S. An, and Doug L. James. Harmonic shells: A

practical nonlinear sound model for near-rigid thin shells. *ACM Transactions on Graphics*, 28(5):119:1–119:10, December 2009.

[24] Jeffrey N. Chadwick and Doug L. James. Animating fire with sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4), August 2011.

[25] Jeffrey N. Chadwick, Changxi Zheng, and Doug L. James. Precomputed acceleration noise for improved rigid-body sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)*, 31(4), August 2012.

[26] A. Chaigne, C. Touzé, and O. Thomas. Nonlinear vibrations and chaos in gongs and cymbals. *Acoustical Science and Technology*, 26(5):403–409, 2005.

[27] D. Chapelle and K.J. Bathe. *The finite element analysis of shells*. Springer, 2003.

[28] Gilsoo Cho, John Casali, and Eunjou Yi. Effect of fabric sound and touch on human subjective sensation. *Fibers and Polymers*, 2:196–202, 2001.

[29] Gilsoo Cho, Chunjeong Kim, Jayoung Cho, and Jiyoung Ha. Physiological signal analyses of frictional sound by structural parameters of warp knitted fabrics. *Fibers and Polymers*, 6:89–94, 2005.

[30] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. *ACM Transactions on Graphics*, 21(3):604–611, July 2002.

[31] M.G. Choi, S. Yong Woo, and H.S. Ko. Real-Time Simulation of Thin Shells. In *Computer Graphics Forum*, volume 26, pages 349–354. Blackwell Publishing Ltd, 2007.

[32] Min Gyu Choi and Hyeong-Seok Ko. Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):91–101, January/February 2005.

[33] F. Cirak and M. Ortiz. Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis. *Internat. J. Numer. Methods Engrg.*, 51:813–833, 2001.

[34] F. Cirak, M. Ortiz, and P. Schroder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg.*, 47:2039–2072, 2000.

[35] Perry Cook. *Real Sound Synthesis for Interactive Applications*. A.K. Peters, 2002.

[36] Stéphane Cotin, Hervé Delingette, and Nicholas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, January/March 1999.

[37] Martin Courshesnes, Pascal Volino, and Nadia Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 137–144, August 1995.

[38] L. Cremer, M. Heckl, and E.E. Ungar. *Structure Borne Sound : Structural Vibrations and Sound Radiation at Audio Frequencies*. Springer, 2nd edition, January 1990.

[39] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling. In *Proc. of ACM SIGGRAPH 2001*, pages 31–36, August 2001.

[40] W. Desmet. Mid-frequency vibro-acoustic modelling: challenges and potential solutions. In *Proceedings of ISMA 2002*, volume II, 2002.

[41] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 732–740, New York, NY, USA, 2003. ACM.

[42] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Synthesizing sound from turbulent field using sound textures for interactive fluid simulation. *Computer Graphics Forum*, 23(3):539–545, September 2004.

[43] S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Synthesizing sound textures through wavelet tree learning. *Computer Graphics and Applications, IEEE*, 22(4):38–48, 2002.

[44] NH Fletcher. The nonlinear physics of musical instruments. *Reports on Progress in Physics*, 62(5):723–764, 1999.

[45] F. Fontana and R. Bresin. Physics-based sound synthesis and control: crushing, walking and running by crumpling sounds. In *Proc. Colloquium on Musical Informatics*, pages 109–114, 2003.

[46] A. Garg, E. Grinspun, M. Wardetzky, and D. Zorin. Cubic shells. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 91–98, 2007.

[47] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[48] Yotam Gingold, Adrian Secord, Jefferson Y. Han, Eitan Grinspun, and Denis Zorin. A Discrete Model for Inelastic Deformation of Thin Shells. Technical report, Courant Institute of Mathematical Sciences, New York University, Aug 2004.

[49] B. L. Giordano. *Sound source perception in impact sounds*. PhD thesis, University of Padova, Italy, 2005.

[50] A. Glassner. The digital ceraunoscope: synthetic thunder and lightning. 2. *Computer Graphics and Applications, IEEE*, 20(3):92 –96, may/jun 2000.

[51] S. Green, G. Turkiyyah, and D. Storti. Subdivision-based multilevel methods for large scale engineering of thin shells. In *Proceedings of ACM Solid Modeling*, pages 265–272, 2002.

[52] E. Grinspun, A. Hirani, M. Desbrun, and P. Schroder. Discrete shells. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 62–67, 2003.

[53] Eitan Grinspun, Petr Krysl, and Peter Schröder. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Trans. on Graphics*, 21(3):281–290, July 2002.

[54] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models. In *Proc. of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 9–20, July 1998.

[55] R. W. Hamming. *Digital Filters*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[56] Michael T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill Higher Education, New York, 2002.

[57] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 229–238, August 1995.

[58] F. B. Hildebrand. *Introduction to Numerical Analysis*. McGraw-Hill, New York, 1956.

[59] Jared Hoberock and Yuntao Jia. Chapter 12: High-Quality Ambient Occlusion. In Hubert Nguyen, editor, *GPU Gems 3*, pages 257–274. Addison-Wesley, 2008.

[60] Ernest Hobson. *The Theory of Spherical and Ellipsoidal Harmonics*. Cambridge University Press, Cambridge, UK.

[61] Hugues Hoppe. New quadric metric for simplifiying meshes with appearance attributes. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 59–66, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[62] Hugues Hoppe and Steve Marschner. Efficient minimization of new quadric metric for simplifying meshes with appearance attributes. Technical Report MSR-TR-2000-64, Microsoft Research, Redmond, WA, June 2000.

[63] P.A. Houle and J.P. Sethna. Acoustic emission from crumpling paper. *Physical Review E*, 54(1):278, 1996.

[64] G. Huang, D. Metaxas, and M. Govindaraj. Feel the "fabric": an audio-haptic interface. In *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 52–61, August 2003.

[65] A.J. Hunt and A.W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 373–376. IEEE, 1996.

[66] Joseph Ivanic and Klaus Ruedenberg. Rotation matrices for real spherical harmonics. direct determination by recursion. *The Journal of Physical Chemistry*, 100(15):6342–6347, 1996.

[67] Doug L. James, Jernej Barbič, and Dinesh K. Pai. Precomputed Acoustic Transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics*, 25(3):987–995, July 2006.

[68] Doug L. James and Kayvon Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics*, 22(3):879–887, July 2003.

[69] Doug L. James and Dinesh K. Pai. ArtDefo: Accurate real time deformable ob-

jects. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 65–72, August 1999.

[70] Doug L. James and Dinesh K. Pai. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Transactions on Graphics*, 21(3):582–585, July 2002.

[71] Doug L. James and Dinesh K. Pai. BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics*, 23(3):393–398, August 2004.

[72] Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Transactions on Graphics*, 24(3):399–407, August 2005.

[73] Jonathan M. Kaldor, Doug L. James, and Steve Marschner. Simulating knitted cloth at the yarn level. *ACM Transactions on Graphics*, 27(3):65:1–65:9, August 2008.

[74] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 279–286, July 2000.

[75] Kevin Karplus and Alex Strong. Digital Synthesis of Plucked-String and Drum Timbres. *Computer Music Journal*, 7(2):43–55, 1983.

[76] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 105–115, July 2008.

[77] Paul G. Kry, Doug L. James, and Dinesh K. Pai. EigenSkin: Real Time Large Deformation Character Skinning in Hardware. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 153–160, July 2002.

[78] P. Krysl, S. Lall, and J. E. Marsden. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering*, 51:479–504, 2001.

[79] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, 26(3):57:1–57:10, July 2007.

[80] C. L. Lawson and R. J. Hanson. *Solving Least Square Problems*. Prentice Hall, Englewood Cliffs, NJ, 1974.

[81] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose Space Deformations: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 165–172, July 2000.

[82] Yaron Lipman, Xiaobai Chen, Ingrid Daubechies, and Thomas Funkhouser. Symmetry factored embedding and distance. *ACM Transactions on Graphics (SIGGRAPH 2010)*, July 2010.

[83] Yi Liu, Amy E. Kerdok, and Robert D. Howe. A nonlinear finite element model of soft tissue indentation. In S. Cotin and D. Metaxas, editors, *Proc. Intl. Symp. Medical Simulation (ISMS 2004), Lecture Notes in Computer Science, vol. 3078*, pages 67–76. Springer-Verlag, 2004.

[84] D. Brandon Lloyd, Nikunj Raghuvanshi, and Naga K. Govindaraju. Sound synthesis for impact sounds in video games. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 55–62 PAGE@7, New York, NY, USA, 2011. ACM.

[85] Charles Loop. *Smooth Subdivision Surfaces Based on Triangles*. PhD thesis, University of Utah, 1987.

[86] Michael Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, Dept. of Computer Science, Univ. of Washington, 1994.

[87] P. Malatkar. *Nonlinear vibrations of cantilever beams and plates*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2003.

[88] D. Marelli, M. Aramaki, R. Kronland-Martinet, and C. Verron. Time–frequency synthesis of noisy sounds with narrow spectral components. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(8):1929–1940, 2010.

[89] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch, and François Sillion. Accurate detection of symmetries in 3d shapes. *ACM Transactions on Graphics*, 25(2):439 – 464, April 2006.

[90] Katsutsugu Matsuyama, Tadahiro Fujimoto, and Norishige Chiba. Real-time sound generation of spark discharge. In *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*, pages 423 –426, 29 2007-nov. 2 2007.

[91] Wojciech Matusik, Matthias Zwicker, and Frédo Durand. Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics*, 24(3):787–794, August 2005.

[92] Roussarie V. Chaigne A. McAdams, S. and B. L. Giordano. The psychomechanics of simulated sound sources: Material properties of impacted plates. *Journal of the Acoustical Society of America*, 128:1401–1413, 2010.

[93] J.H. McDermott, A.J. Oxenham, and E.P. Simoncelli. Sound texture synthesis via filter statistics. In *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA '09. IEEE Workshop on*, pages 297 –300, oct. 2009.

[94] Mark Meyer and John Anderson. Key Point Subspace Acceleration and Soft Caching. *ACM Transactions on Graphics*, 26(3):74:1–74:8, July 2007.

[95] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph.*, 25(3):560–568, July 2006.

[96] Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*, 2012.

[97] William Moss, Hengchin Yeh, Jeong-Mo Hong, Ming C. Lin, and Dinesh Manocha. Sounding liquids: Automatic sound synthesis from fluid simulation. *ACM Trans. Graph.*, 29:21:1–21:13, July 2010.

[98] F. Moussaoui and R. Benamar. Non-linear vibrations of shell-type structures: a review with bibliography. *Journal of Sound and Vibration*, 255(1):161–184, 2002.

[99] A.H. Nayfeh and D.T. Mook. *Nonlinear oscillations*. Wiley-Interscience, 1979.

[100] AH Nayfeh and SA Nayfeh. Nonlinear normal modes of a continuous system with quadratic nonlinearities. *Journal of Vibration and Acoustics*, 117:199, 1995.

[101] Oliver Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer Verlag, December 2000.

[102] James F. O'Brien, Perry R. Cook, and Georg Essl. Synthesizing sounds from physically based motion. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 529–536, August 2001.

[103] James F. O'Brien, Chen Shen, and Christine M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 175–181, July 2002.

[104] Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, and Som H. Yau. Scanning physical interaction behavior of 3d objects. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 87–96, August 2001.

[105] Leevi Peltola, Cumhur Erkut, Perry R. Cook, and Vesa Valimaki. Synthesis of hand clapping sounds. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1021 –1029, 2007.

[106] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, pages 215–222, July 1989.

[107] Cécile Picard, Nicolas Tsingos, and François Faure. Retargetting example sounds to interactive physics-driven animations. In *AES 35th International Conference on Audio for Games*, feb 2009.

[108] J.R. Pierce and S.A. Van Duyne. A passive nonlinear digital filter design which facilitates physics-based sound synthesis of highly nonlinear musical instruments. *The Journal of the Acoustical Society of America*, 101:1120, 1997.

[109] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992. (Gaussian quadrature is discussed in section 4.5).

[110] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics*, 21(3):501–508, July 2002.

[111] Lawrence Rabiner and Biing H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, united states edition, April 1993.

[112] Nikunj Raghuvanshi and Ming C. Lin. Interactive sound synthesis for large scale environments. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 101–108, New York, NY, USA, 2006. ACM.

[113] Nikunj Raghuvanshi, John Snyder, Ravish Mehra, Ming Lin, and Naga Govindaraju. Precomputed wave simulation for real-time sound propagation of dy-

namic sources in complex scenes. *ACM Transactions on Graphics*, 29(4):68:1–68:11, July 2010.

[114] Ganesh Ramanarayanan, James Ferwerda, Bruce Walter, and Kavita Bala. Visual equivalence: Towards a new standard for image fidelity. *ACM Transactions on Graphics*, 26(3):76:1–76:11, July 2007.

[115] Zhimin Ren, Hengchin Yeh, and Ming C. Lin. Synthesizing contact sounds between textured models. In *Proceedings of the 2010 IEEE Virtual Reality Conference*, VR '10, pages 139–146, Washington, DC, USA, 2010. IEEE Computer Society.

[116] Herbert S. Ribner and Dipankar Roy. Acoustics of thunder: A quasilinear model for torturous lightning.

[117] Joshua L. Richmond. Automatic measurement and modeling of contact sounds. Master's thesis, University of British Columbia, April 2000.

[118] C. Roads. *Microsound*. The MIT Press, 2004.

[119] X. Rodet, P. Depall, X. Rodet, and P. Depalle. Spectral envelopes and inverse FFT synthesis. In *Proceedings of the 93rd Audio Engineering Society Convention*, 1992.

[120] Peter Schröder and Wim Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 161–172, August 1995.

[121] Diemo Schwarz. *Data-Driven Concatenative Sound Synthesis*. PhD thesis, Ircam, Centre Pompidou, University of Paris 6–Pierre et Marie Curie, 2004.

[122] Diemo Schwarz. Current research in concatenative sound synthesis. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 9–12, 2005.

[123] Jaewoo Seo, Geoffrey Irving, J. P. Lewis, and Junyong Noh. Compression and direct manipulation of complex blendshape models. *ACM Trans. Graph.*, 30(6):164:1–164:10, December 2011.

[124] Xavier Serra and III Smith, Julius. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):pp. 12–24, 1990.

[125] Ahmed Shabana. *Dynamics of Multibody Systems*. Cambridge, 3rd edition, 2005.

[126] Ahmed A. Shabana. *Theory of Vibration, Volume II: Discrete and Continuous Systems*. Springer–Verlag, New York, NY, 1990.

[127] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics*, 22(3):382–391, July 2003.

[128] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21(3):527–536, July 2002.

[129] Peter-Pike Sloan, Ben Luna, and John Snyder. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics*, 24(3):1216–1224, August 2005.

[130] G. Strobl, G. Eckel, D. Rocchesso, and S. le Grazie. Sound texture modeling: A survey. In *Proceedings of the 2006 Sound and Music Computing (SMC) International Conference*, pages 61–5, 2006.

[131] Tapio Takala and James Hahn. Sound rendering. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 211–220, July 1992.

[132] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, December 1988.

[133] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, pages 205–214, July 1987.

[134] Demetri Terzopoulos and Andrew Witkin. Physically Based Models with Rigid and Deformable Components. *IEEE Computer Graphics & Applications*, 8(6):41–51, November 1988.

[135] C. Touzé, O. Thomas, and A. Chaigne. Hardening/softening behaviour in non-linear oscillations of structural systems using non-linear normal modes. *Journal of Sound and Vibration*, 273(1-2):77–101, 2004.

[136] Nicolas Tsingos, Thomas Funkhouser, Addy Ngan, and Ingrid Carlbom. Modeling acoustics in virtual environments using the uniform theory of diffraction. In

Proceedings of ACM SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Series, pages 545–552, August 2001.

[137] A.F. Vakakis. *Normal modes and localization in nonlinear systems*. Kluwer Academic Publishers, 2001.

[138] K. van den Doel and D. K. Pai. Synthesis of shape dependent sounds with physical modeling. In *Intl Conf. on Auditory Display*, Xerox PARC, Palo Alto, November 1996.

[139] Kees van den Doel, Dave Knott, and Dinesh K. Pai. Interactive simulation of complex audiovisual scenes. *Presence: Teleoper. Virtual Environ.*, 13(1):99–111, 2004.

[140] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. FoleyAutomatic: Physically Based Sound Effects for Interactive Simulation and Animation. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 537–544, August 2001.

[141] Grace Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Philadephia, PA, 1990.

[142] Andrew Witkin and William Welch. Fast Animation and Control of Nonrigid Structures. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, pages 243–252, August 1990.

[143] Changxi Zheng and Doug L. James. Harmonic fluids. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 37:1–37:12, New York, NY, USA, 2009. ACM.

[144] Changxi Zheng and Doug L. James. Rigid-body fracture sound with precomputed soundbanks. *ACM Transactions on Graphics*, 29(4):69:1–69:13, July 2010.

[145] Changxi Zheng and Doug L. James. Toward high-quality modal contact sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4), August 2011.

[146] E. Zwicker and H. Fastl. *Psychoacoustics*. Springer-Verlag, 1990.